PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-155010
(43)Date of publication of application : 09.06.1998

(51)Int.Cl. H04L 29/08
G06F 9/46
G06F 13/00
H04L 13/08

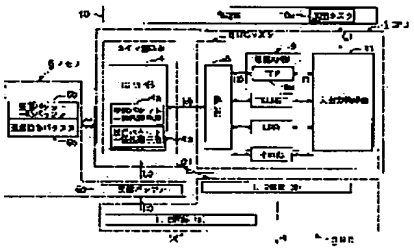
(21)Application number : 09-263779
(22)Date of filing : 29.09.1997

(71)Applicant : OKI DATA:KK
(72)Inventor : SUDO HIDEO

(30)Priority
Priority number : 08258957 Priority date : 30.09.1996 Priority country : JP

(54) PACKET PROCESSING METHOD AND NETWORK ARCHITECTURE

(57)Abstract:
PROBLEM TO BE SOLVED: To reduce an overhead of an OS by processing a network layer and a transport layer with one function as a reception packet simultaneous processing means so as to process all packets received by a reception buffer through one start of protocol processing.
SOLUTION: A reception packet simultaneous processing means 4a of a TCP/IP section 4 is started at timer interrupt and processes simultaneously protocol processing relating to a network layer and a transport layer of a packet received in a reception buffer 5 within an interrupt time to store the received packet assigned in a memory 5 to a temporary buffer 5b and to post it to an IFC task 6. Furthermore, a transmission packet simultaneous processing means 4b sends a reply packet or the like stored in a transmission wait buffer 5c assigned in the memory 5 to a network within the interrupt time. The TCP /IP section 4 is operated by using a TCP and an IP as one task without conducting them as separate tasks.



LEGAL STATUS

[Date of request for examination] 29.01.2001
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平10－155010

(43)公開日　平成10年(1998)６月９日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｈ０４Ｌ　29/08		Ｈ０４Ｌ　13/00	３０７Ｚ
Ｇ０６Ｆ　9/46	３６０	Ｇ０６Ｆ　9/46	３６０Ｆ
13/00	３５３	13/00	３５３Ｃ
Ｈ０４Ｌ　13/08		Ｈ０４Ｌ　13/08	

審査請求　未請求　請求項の数14　ＯＬ　（全 24 頁）

(21)出願番号　特願平9－263779

(22)出願日　平成９年(1997)９月29日

(31)優先権主張番号　特願平8－258957

(32)優先日　平８(1996)９月30日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　591044164
株式会社沖データ
東京都港区芝浦四丁目11番地22号

(72)発明者　須藤　日出夫
東京都港区芝浦４丁目11番地22号　株式会社沖データ内

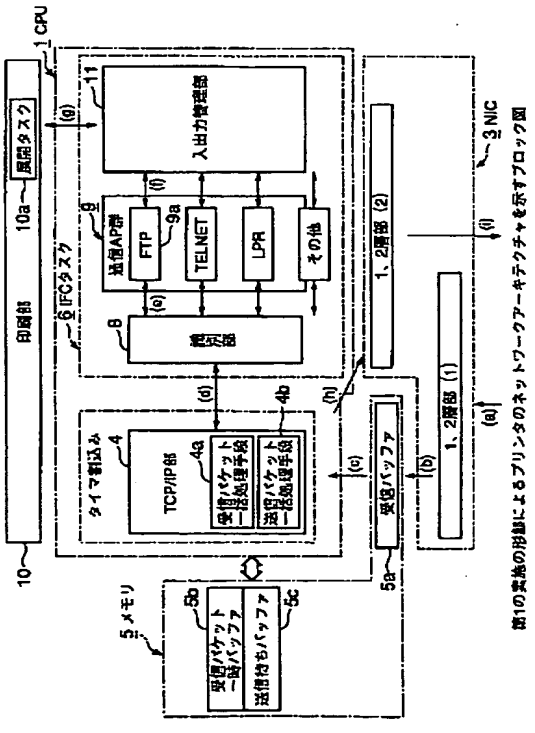
(74)代理人　弁理士　大西　健治

(54)【発明の名称】　パケット処理方法とネットワークアーキテクチャ

(57)【要約】

【目的】　ネットワークアーキテクチャに組み込まれるネットワーク層／トランスポート層の負荷を軽減し、より効率良く動作するパケット処理方法とネットワークアーキテクチャとを提供する。

【構成】　ネットワークからノードの受信バッファ５ａにパケットを受信し、応答パケットをネットワークに送信するＮＩＣ（Network Interface Controller）３と、一定時間毎に入るタイマ割込みを使用し、割込み時間内に受信バッファ５ａに受信される全パケットのネットワーク層とトランスポート層とに関するプロトコルを１回の起動で一括処理し、それぞれのコネクションに属する最後の受信パケットに対する応答パケットを割込み中にネットワークに一括して送信するＴＣＰ／ＩＰ部５とを備える。



第1の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図

【特許請求の範囲】

【請求項１】　ネットワークのノード間にコネクションを確立し、受信バッファに受信したパケットに階層的に付加されたプロトコルを下位層から上位層に向かって順に処理するパケット処理方法において、
少なくともネットワーク層、トランスポート層、セッション層、プレゼンテーション層、アプリケーション層に関するプロトコルの処理をそれぞれ一つのタスクにし、ネットワーク層、トランスポート層を一つの機能とし、セッション層、プレゼンテーション層、アプリケーション層を一つの機能とし、その機能に処理の優先度を持たせて動作するようにしたことを特徴とするパケット処理方法。

【請求項２】　ネットワークのノード間にコネクションを確立し、受信バッファに受信したパケットに階層的に付加されたプロトコルを下位層から上位層に向かって順に処理するネットワークアーキテクチャにおいて、
一定時間毎に入るタイマ割込みを使用し、上記受信バッファに受信されたパケットのネットワーク層とトランスポート層とに関するプロトコルを割込み中に一括処理して蓄積し、プロトコルの内容から上位層のプロトコル処理手段を指定通知する受信パケット一括処理手段と、
それぞれのコネクションに属する最後の受信パケットに対する応答パケットを上記割込み中にネットワークに一括して送信する送信パケット一括処理手段とを備えたことを特徴とするネットワークアーキテクチャ。

【請求項３】　上記送信パケット一括処理手段は、上記受信パケット一括処理手段が受信パケットを一括処理中に上位層の処理手段から発生した応答パケット送信要求を、次のタイマ割込み時にまとめて送信を行い、それに対する受信側からの応答のタイムアウトをタイマ割込み回数でカウントするタイマ処理手段を備えた請求項２記載のネットワークアーキテクチャ。

【請求項４】　ネットワークのノード間にコネクションを確立し、受信バッファに受信したパケットに階層的に付加されたプロトコルを下位層から上位層に向かって順に処理するネットワークアーキテクチャにおいて、
メッセージに基づいて呼び出した処理手段に起動を駆けるオペレーティングシステム機能によるメッセージ処理手段と、メッセージ処理手段により呼び出されて受信バッファから受信パケットを取り出し、データリンク層、ネットワーク層、トランスポート層に関するプロトコルを処理して所定量に達するまで蓄積する第１のパケット処理手段と、セッション層、プレゼンテーション層、アプリケーション層に関するプロトコルを処理して上位層に出力する第２のパケット処理手段と、第１のパケット処理手段から指定された第２のパケット処理手段を呼び出すメッセージをメッセージ処理手段に出力し、メッセージ処理手段からの呼び出しに応じて第２のパケット処理手段を選択するソケット処理手段とを設けたインターフェース制御手段と、
パケット受信時に起動し、受信パケットを受信バッファに格納する毎にメッセージをインターフェース制御手段に出力する受信割り込み手段とを備えたことを特徴とするネットワークアーキテクチャ。

【請求項５】　上記第１のパケット処理手段は、上記受信割り込み手段のメッセージに基づいて呼び出され、受信バッファから受信パケットを取り出してデータリンク層のプロトコルを処理するネットワークドライバを有する請求項４記載のネットワークアーキテクチャ。

【請求項６】　ネットワークのノード間にコネクションを確立し、受信バッファに受信したパケットに階層的に付加されたプロトコルを下位層から上位層に向かって順に処理するネットワークアーキテクチャにおいて、
メッセージに基づいて呼び出した処理手段に起動を駆けるオペレーティングシステム機能によるメッセージ処理手段と、ネットワーク層、トランスポート層に関するプロトコルを処理して所定量に達するまで蓄積する第１のパケット処理手段と、セッション層、プレゼンテーション層、アプリケーション層に関するプロトコルを処理して上位層に出力する第２のパケット処理手段と、第１のパケット処理手段から指定された第２のパケット処理手段を呼び出すメッセージをメッセージ処理手段に出力し、メッセージ処理手段からの呼び出しに応じて第２のパケット処理手段を選択するソケット処理手段とを有するインターフェース制御手段と、
パケット受信時に起動し、受信パケットを受信バッファに格納する毎にメッセージをインターフェース制御手段に出力する受信割り込み手段と、
受信割り込み手段のメッセージに基づいてインターフェース制御手段のメッセージ処理手段により呼び出され、データリンク層に関するプロトコルを処理して第１のパケット処理手段を呼び出すネットワークドライバとを備えたことを特徴とするネットワークアーキテクチャ。

【請求項７】　カウンタがカウンタアップする毎にメッセージを上記インターフェース制御手段のメッセージ処理手段に出力するインターバルタイマと、インターバルタイマのメッセージに基づいて呼び出され、上記第１のパケット処理手段により蓄積されたパケット量を監視するタイマ処理手段を上記第１のパケット処理手段に設けた請求項４記載、又は請求項６記載のネットワークアーキテクチャ。

【請求項８】　上記タイマ処理手段は、上記第１のパケット処理手段がプロトコル処理中にタイマ処理を実行し、応答パケットの送信処理等のプライオリティの低い処理に負荷係数を設定し、一定の負荷に達した時点でタイマ処理を中断し、次回のタイマ処理でその続きを実行することにより、１回のタイマ処理時に処理する負荷を平均化させる請求項７記載のネットワークアーキテクチャ。

【請求項９】　上記タイマ処理部は、コネクションを増減する際に参照するテーブルを記憶するテーブル記憶手段と、前回起動時の受信バッファの使用量を格納しておく第１の受信バッファ使用量記憶手段と、上記メッセージ処理手段に起動される毎に受信バッファの使用量をセンスして第２の受信バッファ使用量記憶手段に格納するバッファ量センス部と、第１の受信バッファ使用量記憶手段の内容と第２の受信バッファ使用量記憶手段の内容とテーブル記憶手段のテーブルとを参照して増減するコネクション数を判断する判断部と、判断部の出力に基づいて許容コネクション数を設定するコネクション数設定部とを備えた請求項７記載のネットワークアーキテクチャ。

【請求項１０】　上記データバッファに格納されたデータを処理するプライオリティの低い処理手段に設けられた負荷監視部と、負荷監視部からの負荷通知を記憶する負荷通知記憶手段と、負荷通知記憶手段の内容が高負荷通知ならば上記インターバルタイマに設けられたカウンタのタイマ間隔値を大きくし、否ならばカウンタのタイマ間隔値を小さくするタイマ間隔値変更手段とを備えた請求項７記載のネットワークアーキテクチャ。

【請求項１１】　上記メッセージ処理手段はメッセージを格納するＥＣＢ（Event Control Block ）を一つ有し、ＥＣＢとソケット処理手段の上位側待ち行列とを参照し、上位側待ち行列の登録メッセージ数が所定数以上の場合には上位側待ち行列を処理する請求項４記載、又は請求項６記載のネットワークアーキテクチャ。

【請求項１２】　上記第２のパケット処理手段はデータのクライアントＩＤを格納する共通テーブルを有し、データを出力する際にクライアントＩＤを格納し、上位層からデータ送信要求を受ける際には共通テーブルに基づいてクライアントＩＤを解析し、同一コネクションを通じてパケットを送受信する請求項４記載、又は請求項６記載のネットワークアーキテクチャ。

【請求項１３】　第１のパケット処理手段が前回の受信パケットに対する応答パケットを送信してから今回の受信パケットを受信までのパケット転送時間を算出するパケット転送時間算出手段と、第２のパケット処理手段から単位時間にデータを上位層に転送する転送データ数を計数する転送データ数計数手段と、パケット転送時間と転送データ数と現在の受信バッファ空容量とから次回のパケットブロック転送において受信可能なデータ容量を算出する受信可能データ容量算出手段とを設けた請求項４記載、又は請求項６記載のネットワークアーキテクチャ。

【請求項１４】　上記受信割り込み手段は、上記受信バッファの「常時使用エリア」と「常時空きエリア」とに受信したパケットの情報を格納する共通テーブルを有し、パケットを受信する毎に共通テーブルを参照して「常時空きエリア」に受信したパケットが送信側から受信側に送信される受信可能確認用ポーリングパケットの場合には応答パケットを作成して送信し、受信可能確認用ポーリングパケット以外の場合にはそのパケットを破棄する請求項４記載、又は請求項６記載のネットワークアーキテクチャ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明はネットワークのノード間にコネクションを確立し、受信バッファに受信したパケットに階層的に付加されたプロトコルを下位層から上位層に向かって順に処理するパケット処理方法とネットワークアーキテクチャとに関する。

【０００２】

【従来の技術】従来、ネットワークのノードに組込まれるＯＳＩ(Open System Interconnection) 環境の標準ネットワークアーキテクチャは、プロトコルを階層的に並べた構造を採っており、ＩＳＯ(International Organization for Standardization)等により、各層の位置付けや、機能、層間のインタフェースなどが細かく規定されている。

【０００３】ＩＳＯが規定したＯＳＩ参照モデルには、下位から物理層、データリンク層、ネットワーク層、トランスポート層、セッション層、プレゼンテーション層、アプリケーション層の７層構造になっている。

【０００４】ネットワークに送り出されるパケット形式のデータには、それぞれ、各層に関する制御情報であるプロトコルがヘッダとして付加されており、各ノードのネットワークアーキテクチャはネットワークからパケットを受信してパケットのヘッダに階層的に含まれるプロトコルを各層で順次処理してパケットに含まれるデータをデータバッファに格納するとともに、受信パケットに対する応答パケットをネットワークに送信する。

【０００５】例えば、インターネットプロトコル層（以後ＩＰと記す）はネットワーク層に位置し、主に一つ又は複数の通信網を介して中継を行い、始点となるノードから終点となるノードまでのデータ通信を行う。

【０００６】また、ＩＰの上位層（トランスポート層）としてのトランスファコントロールプロトコル層（以後ＴＣＰと記す）は、経路選択や中継機能に関与せず、始点及び終点となるノードのさらに終端間における透過的な両方向同時転送機能を提供する。

【０００７】ＴＣＰをネットワークアーキテクチャに組込むには、一つのコネクション（データを転送するための論理的な通信路）に対する処理を一つのＴＣＰタスクで行せるようにし、Ｎ個のコネクションに対して（Ｎ＋１）個のＴＣＰタスクを起動させる。ＴＣＰタスクの残りの一つはマスタとして次のコネクションの確立用として待機している。

【０００８】ＩＰは、１つのタスクが、受信した受信パケットのヘッダ処理を順次処理して対応するＴＣＰタス

クを識別し、待機しているＴＣＰタスクに預けていく処理を行っている。よって確立するコネクション数とは関係なく常に１つのタスクとして存在する。

【０００９】各ＴＣＰタスクはそれぞれコネクションを確立したクライアントタスクから受信した受信パケットを処理し、相手のクライアントタスクに応答パケットを送信する。このとき送信する応答パケットはただ一つ存在するＩＰタスクを通る必要があるため、各ＴＣＰタスクはＩＰタスクの使用権を獲得する。

【００１０】ＩＰタスクはＴＣＰタスクまたはデータリンク層タスクからの呼び出し（以後ＰＯＳＴと記す）を待つ。ＰＯＳＴとは、マルチタスク環境下のタスク間で処理を移行する時に使用するオペレーションシステム（以後ＯＳと記す）のシステムコールである。

【００１１】ここで、簡単にＩＰタスクとＴＣＰタスクとの動作を説明する。データリンク層タスクは、クライアントより最初のパケットを受信すると、そのヘッダからＩＰタスクを識別し、待機しているＩＰタスクに対してＰＯＳＴを行う。

【００１２】ＩＰタスクは受信パケットのヘッダからネットワーク層に関するプロトコル処理を行い、ヘッダからＴＣＰタスクを識別し、待機しているＴＣＰマスタタスクに対してＰＯＳＴを行う。

【００１３】ＴＣＰマスタタスクは受信パケットの内容がコネクションの開設要求と判断すると、パケットを処理し、クライアントに対して応答パケットを送信する。通常クライアントからの最初のパケットはコネクションの開設を要求するパケットなので、クライアントへ応答パケットを送信した後はタイマを動作させてスリーウェイハンドシェークの最後のパケットを待つ。

【００１４】ＴＣＰマスタタスクは、タイマアップする前にクライアントからハンドシェークパケットを受信すると、コネクションの確立を認識する。また、コネクション確立後の受信パケットがセッション層以上のサービスの要求がある場合には、新たにマスタタスクを生成しＰＯＳＴを行う。本例では、ＦＴＰ（File TransferProtocol）マスタタスクを起動する。

【００１５】ＦＴＰマスタタスクは、クライアントとの間に制御コネクションを確立すると、fork()関数を用い自分自身のコピー（ＦＴＰタスク）を作成する。ＦＴＰタスクは、ＴＣＰマスタタスクに対してポートの割り当てを行う。

【００１６】ＴＣＰマスタタスクは、fork()関数を用い自分自身のコピー（ＴＣＰタスク）を作成する。ＴＣＰタスクは、ＦＴＰタスクとの間にデータコネクションを確立し、以後クライアントから切断されるまでそのコネクションを管理する。

【００１７】ＴＣＰマスタタスクは次のクライアントからのコネクションの開設要求に備えて待ち状態となる。

【００１８】他方、ＴＣＰタスクはパケットの受信毎にトランスポート層の処理を行い、ＦＴＰタスクに対してＰＯＳＴを行う。ＦＴＰタスクは、ＴＣＰタスクから受信したデータを受け取り、データ中に含まれるクライアントからのコマンドを実行する。

【００１９】以上説明した組込み処理は、完全にプリエンプティブに動作するラウンドロビン方式のＯＳの組込みを前提としている。

【００２０】

【発明が解決しようとする課題】従来のネットワークアーキテクチャにあっては、複数のコネクションをそれと同数のＴＣＰタスクで処理するため、コネクションを処理するプロセスを動的に複数コピーする機能（上述したfork()関数）が必要となる。この機能は１つのプログラムを複数のタスクで共有し実行させるため、タスク間の同期を取るラウンドロビン方式の完全にプリエンプティブに動作する高機能なＯＳを必要とするという問題点があった。

【００２１】また、各層のプロトコル処理を別々のタスクとして動作させるとともに、トランスポート層以上のタスクはコネクションの確立に応じてタスクがコピーされるので、ＯＳ上で動作するタスクの数が増え、ＯＳのオーバヘッド量が増加するという問題点もあった。

【００２２】また、プロトコル処理における各プロセスをすべて別々のタスクが行うことにより、メモリ上の作業領域が増大するという問題点もあった。

【００２３】本発明はネットワークアーキテクチャに組み込まれるネットワーク層／トランスポート層の負荷を軽減し、より効率良く動作するパケット処理方法とネットワークアーキテクチャとを提供することを目的とする。

【００２４】

【課題を解決するための手段】上記目的を達成するために本発明のネットワークアーキテクチャにおいては、一定時間毎に入るタイマ割込みを使用し、受信バッファに受信されたパケットのネットワーク層とトランスポート層とに関するプロトコルを割込み中に一括処理して蓄積し、プロトコルの内容から上位層のプロトコル処理手段を指定通知する受信パケット一括処理手段と、それぞれのコネクションに属する最後の受信パケットに対する応答パケットを割込み中にネットワークに一括して送信する送信パケット一括処理手段とを備える。

【００２５】また、同じ目的を達成するために本発明のパケット処理方法においては、少なくともネットワーク層、トランスポート層、セッション層、プレゼンテーション層、アプリケーション層に関するプロトコルの処理をそれぞれ一つのタスクにし、ネットワーク層、トランスポート層を一つの機能とし、セッション層、プレゼンテーション層、アプリケーション層を一つの機能とし、その機能に処理の優先度を持たせて動作するパケット処理方法とし、具体的手段として、メッセージに基づいて

呼び出した処理手段に起動を駆けるオペレーティングシステム機能によるメッセージ処理手段と、メッセージ処理手段により呼び出されて受信バッファから受信パケットを取り出し、少なくともネットワーク層、トランスポート層に関するプロトコルを処理して所定量に達するまで蓄積する第１のパケット処理手段と、セッション層、プレゼンテーション層、アプリケーション層に関するプロトコルを処理して上位層に出力する第２のパケット処理手段と、第１のパケット処理手段から指定された第２のパケット処理手段を呼び出すメッセージをメッセージ処理手段に出力するとともに上位側待ち行列に登録し、メッセージ処理手段からの呼び出しに応じて上位側待ち行列を参照し、第２のパケット処理手段を呼び出すソケット処理手段とを備える。

【００２６】

【発明の実施の形態】本発明の実施の形態について図面を参照しながら説明する。尚、各図面に共通な要素には同一符号を付す。

第１の実施の形態

図１は第１の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図であり、単一な中央処理装置１（以後ＣＰＵ１と記す）上で動作する割込み処理及びタスクを示す。

【００２７】尚、ノードとして、図２に示すように、対象を回線に接続されたネットワークプリンタ２（以後プリンタ２と記す）にしぼり説明する。

【００２８】ネットワークアーキテクチャは、ネットワーク接続用ハードウェアとしてのＮＩＣ（Network Interface Controller）３と、トランスポート層及びネットワーク層のプロトコル処理手段としてのＴＣＰ／ＩＰ部４と、クライアントとのデータの送受信を制御し、メモリ５上に割り当てられた受信バッファ５ａを管理するＩＦＣタスク６とからなる。

【００２９】ＮＩＣ３はネットワークからのパケットの受信時に起動し、ＣＰＵ１に割り込みをかけて受信パケットを受信バッファ５に受信し、受信パケットに対する応答パケット等を送信する時に使用される。

【００３０】ＴＣＰ／ＩＰ部４はタイマ割り込み時に起動し、割り込み時間内に受信バッファ５に受信されてある受信パケットのネットワーク層とトランスポート層とに関するプロトコル処理を一括処理してメモリ５上に割り当てられた受信パケット一時バッファ５ｂに格納してＩＦＣ（Interface Control ）タスク６にＰＯＳＴを行う受信パケット一括処理手段４ａと、メモリ５上に割り当てられた送信待ちバッファ５ｃに格納されてある応答パケット等を割り込み時間内にネットワークに送信する送信パケット一括処理手段４ｂとが設けてある。ＴＣＰ／ＩＰ部４は、ＴＣＰとＩＰとを別タスクとせず一つのタスクとして動作する。

【００３１】ＩＦＣタスク６は、ＴＣＰ／ＩＰ部４により指定された通信アプリケーションを識別する通信アプリケーション識別部８と、受信パケットのセッション層、プレゼンテーション層、アプリケーション層に関するプロトコルを処理する通信アプリケーション群９と、受信パケットに含まれる受信データを通信アプリケーション群９から入力して印刷部１０に出力する入出力管理部１１とからなる。

【００３２】図３は図１に示したネットワークアーキテクチャの処理時のタイムチャートである。ＴＣＰ／ＩＰ部４は時間ｔ経過する毎にタイマ割り込みに入るように設定されており、割り込み時間内にＴＣＰ／ＩＰ部４のプロトコル処理を行う。時間ｔは割り込み処理時間に比べて大きく設定されてあり、ＴＣＰ／ＩＰ部４が割り込み処理を終了するとＰＯＳＴを行い、以後、次のタイマ割り込みに入るまでＮＩＣ３によるパケット送受信処理、ＩＦＣタスク６によるパケット処理、印刷部１０による印刷処理等が行われる。

【００３３】ＴＣＰ／ＩＰ部４はタイマ割り込みにより起動すると、受信バッファ５に受信済みの全パケットを割り込み時間内に処理し、最後のパケットに対する応答パケットを作成して送信待ちバッファ７ｂに格納する。たとえば、受信バッファ５にクライアントＡから３つのパケットA1 ～A3 を受信してあるとすると、１回のタイマ割り込み処理で３つのパケットA1 ～A3 を処理し、パケットA3 に対する応答パケットを作成して送信待ちバッファ７ｂに格納する。

【００３４】また、複数のクライアントからパケットを受信した場合にも同様に、各クライアントから受信した最後のパケットに対して応答パケットを作成して送信待ちバッファ７ｂに格納する。つまり、クライアントＡから３つのパケットA1 ～A3を、クライアントＢから１つのパケットB1 を、クライアントＣから２つのパケットC1 、C2 を受信バッファ５に受信していた場合、ＴＣＰ／ＩＰ部４は、クライアントＡにはパケットA3 に対する応答パケットを、クライアントＢにはパケットＢ１ に対する応答パケットを、クライアントＣにはパケットC2 に対する応答パケットをそれぞれ作成して送信待ちバッファ７ｂに格納する。

【００３５】送信処理はＮＩＣ３に対して送信に必要な情報をセットする前処理、ＮＩＣ３が行う送信処理、送信完了後行う後処理からなる。

【００３６】送信パターンの１つ目は、ＴＰＣ／ＩＰ部４から応答される場合で、ＴＰＣ／ＩＰ部４とのコネクション確立シーケンス時の応答、データパケットに対する応答、ARP(Address Resolution Protocol)における物理アドレスの問い合わせに対する応答等、受信パケットを受信した直後に発生する送信処理であり、上述したタイマ割り込み処理中に行う。

【００３７】即ち、データパケットのストリームに対する単なる応答である場合には、上述した通りに直ちに送

信を行わず、受信済みの全パケットの処理が終了した後に応答パケットを作成する。その他の応答の場合には、直ちに送信用のパケットを作成して送信待ちバッファ７ｂに格納し、タイマ割り込み処理の最後にまとめて送信要求をＮＩＣ３に出す。

【００３８】こうすることにより、送信パケットに対する応答を受信するまでのタイムアウト時間(ターンアラウンド時間）を一括に管理する（以後Ｍａｘ　Ｔｉｍｅｏｕｔ処理と記す）ことができ、タイムアウト処理にかかる負荷が減少する。

【００３９】本実施の形態によるアーキテクチャ組込みにおけるタイマ処理は、各コネクションを管理する情報とともにタイマ情報をグローバルテーブルとして管理する。送信パケットのタイムアウト値、タイムアウトカウンタ等をテーブル中に保存し、タイマ割り込み中でカウンタをデクリメントし、タイマ割り込みの入る回数でタイムアウト時間を管理する。タイムアウト値はＴＣＰ／ＩＰ部４のタイマ割り込みが入る時間ｔの倍数で定義し、ｔの倍数単位の増減を行う。

【００４０】送信パターンの２つ目として、印刷部１０のプリンタファームウェアや通信アプリケーション群９の通信アプリケーション等から任意の時間に発生する送信処理がある。

【００４１】従来のアーキテクチャ組込みによる送信では、ＴＣＰタスク及びＩＰタスクは、ＩＦＣタスク６から送信要求を受けると、図４（Ａ）に示すように、直ちに送信処理を行い、ＮＩＣ３へ送信処理を要求した。

【００４２】しかし、アプリケーションの送信では、タイムアウトが存在しないか、存在しても非常に長いので緊急に送信する必要はない。従って、本発明ではＩＦＣタスク６の送信要求を一定時間蓄積し、一括して送信する方法を採る。

【００４３】通信アプリケーション関数内及びプリンタファームウェア内で送信要求が発生すると、通信アプリケーション関数が、特定のテーブルに対して送信要求を書き込む。実際の送信は、図４（Ｂ）に示すように、ＴＣＰ／ＩＰ部４が次に入るタイマ割り込み中に通信アプリケーションによってセットされたテーブル情報を参照して行う。

【００４４】また、タイマ割り込み時に受信パケットが受信バッファ５内に存在する時は、受信パケットの処理を優先して行い、その応答パケットと一緒に送信要求をＮＩＣ３に対して行う。

【００４５】次に動作について説明する。クライアントよりランダムに送られてくるパケットは、図１に矢印（ａ）、（ｂ）で示すように、第１、２層部のＮＩＣ３によりリアルタイムに受信される。ＮＩＣ３はパケットを受信する毎にＣＰＵ１に割り込みをかけて、割り込み処理によりパケットを受信バッファ５に格納する。

【００４６】ＴＣＰ／ＩＰ部４は、時間ｔ毎に入るタイマ割り込み中にネットワーク層及びトランスポート層のプロトコル処理を、図１に矢印（ｃ）で示すように、その時受信バッファ５に受信してある全パケットについて行い、処理済みのパケットをパケット一時バッファ７ａに格納する。

【００４７】一連の受信データストリームにおける最終パケットを受信すると、ＴＣＰ／ＩＰ部４は、ヘッダから上位層プロトコル(通信アプリケーション）を識別し、それを通知するフラグをセットして、図１に矢印（ｄ）で示すように、ＩＦＣタスク６に対してＰＯＳＴを行う。

【００４８】フラグはグローバルな変数で、ＴＣＰ／ＩＰ部４が受信データストリームを渡すべき上位層(通信アプリケーション）を指定するために用い、ＴＣＰ／ＩＰ部４からＩＦＣタスク６に対して受け渡される。受け渡し方法は、ＯＳのメッセージパッシング機能を用いる。

【００４９】通信アプリケーションは、すべてＩＦＣタスク６の関数として組み込まれ、指定された上位層が、例えば、ＦＴＰ９ａとすると、図１に矢印（ｅ）で示すように、識別部８はＦＴＰ関数をコールする。

【００５０】ＦＴＰ９ａは、ＴＣＰ／ＩＰ部４が受信したデータストリームを受信パケット一時バッファ７ａから受け取り、内容を処理する。必要であれば、図１に矢印（ｆ）で示すように、入出力管理部１１へデータを渡す。入出力管理部１１へ渡すデータは、コネクション別になっているので、コネクション別のキュー(以後、ＩＦＣ－ＥＤＩＴ間受信キューと記す）を作成する。

【００５１】ＩＦＣ－ＥＤＩＴ間受信キューは、ＩＦＣタスク６の通信アプリケーションＦＴＰ９ａから入出力管理部１１を介し、印刷部１０のＥＤＩＴ（編集）タスク１０ａへ印刷データ、その他を渡す待ち行列である。印刷部１０のＥＤＩＴタスク１０ａは、図１に矢印（ｇ）で示すように、ＩＦＣ－ＥＤＩＴ間受信キューからデータを受け取り編集処理を行う。

【００５２】ここで、従来の方法（Ａとする）による受信処理時間ｒＡと、本実施の形態方法（Ｂとする）による受信処理時間ｒＢとの差Ｒｅｃを以下に示す。

【００５３】まず、処理をＴＣＰ／ＩＰ部４に、時間をｔ内に限定し、パケット受信時間を、パケット処理時間とＯＳの処理時間に分けて求める。単位時間に受信するパケットの数をｎ、１パケットを処理する時間をＥとすると一定時間ｔにおけるパケットの処理時間Ｅ（Ａ，Ｂ）は、Ａ，Ｂ共に、

Ｅ（Ａ，Ｂ）＝Ｅ×ｔ×ｎ・・・・・・・・・・・（１）

また、ＯＳの処理時間は、タスク生成時間とタスクスイッチ時間との和とすると、fork()関数等により子タスクを作成し実行させる時間をＦ、同時にアクセスする平均コネクション数をＮとすると一定時間ｔにおけるタスク生成処理（コネクション確立毎生成）時間Ｆ（Ａ）は、

Aのみ該当し、

$$F(A)=F\times N \cdots\cdots\cdots\cdots\cdots (2)$$

タスクスイッチに要する時間（ＰＯＳＴされてからタスクが起動するまでの時間）において、従来の組込みにおいて対象となるＯＳをＰＡ、割込みによるマルチタスクＯＳをＰＢとすると１パケット受信に対しＩＰタスク、ＴＣＰタスクがそれぞれ起動するため一定時間ｔにおけるタスクチェンジ処理時間Ｐ（Ａ）、Ｐ（Ｂ）は、Ａでは、１パケット受信毎にＩＰタスク、ＴＣＰタスクが呼び出されるので、

$$P(A)=2\times PA\times t\times n \cdots\cdots\cdots (3)$$

Ｂでは、ｔ時間内に１度ＴＣＰ／ＩＰ部４が呼ばれるので、

$$P(B)=PB \cdots\cdots\cdots\cdots\cdots (4)$$

従来方法の一定時間ｔにおける受信処理に要する時間ｒＡは式（１）（２）（３）より、

$$rA=E(A,B)+F(A)+P(A)$$
$$=Etn+FN+2(PA)tn \cdots (5)$$

タイマ割り込みが起動し、割り込み内の処理が起動するまでの時間をＩとすると、本方法の一定時間ｔにおける受信処理に要する時間ｒＢは、（１）（４）より、

$$rB=E(A,B)+P(B)+I$$
$$=Etn+PB+I \cdots\cdots\cdots (6)$$

となるから、それらの差Ｒｅｃは式（５）、（６）より次の式で示すことができる。

【００５４】

$$Rec=rA-rB$$
$$=FN+2(PA)tn-PB-I \cdots (7)$$

従って、本組込み方法により一定時間ｔにおけるパケットの受信時間をＦＮ＋２（ＰＡ）ｔｎ－ＰＢ－Ｉだけ短縮することができる。

【００５５】また、従来の組込み方法では、受信側ノードは１パケットを受信するとＩＰタスクを経由し、直ちに制御をＴＣＰタスクへ移す。そしてＴＣＰタスクが１パケットの処理を終了すると、制御を他のタスクへ移さなければならないので、後続のパケットの受信を待って、それらをまとめて応答することは難しい。そのため、１パケット受信する毎に応答パケットをクライアントに返していた。

【００５６】従来の方法の応答処理時間ＳＡと本実施の形態による方法の応答処理時間ＳＢとの差Ｒｅｓは次のように求めることができる。

【００５７】

$$SA=t\times n\times R+t\times n\times PA+t\times n\times S \cdots (8)$$
$$SB=N\times R+S \cdots\cdots\cdots\cdots\cdots (9)$$

式（８）（９）より、

$$Res=SA-SB$$
$$=tnR+tn(PA)-NR+(tn-1)S \cdots (10)$$

ｎは単位時間に受信するパケット数、ｔは本実施の形態による組込みで採用するタイマ割込みの間隔、ＰＡは従来の組込みにおいて対象となるＯＳのタスクスイッチ時間、ＰＢは割込みよるマルチタスクＯＳのタスクスイッチ時間、Ｎは同時にアクセスする平均コネクション数、Ｒは１パケットの応答処理に要する時間、ＳはＮＩＣ３への送信要求処理時間と送信完了割り込み処理時間の和である。従って、本実施の形態による組込み処理によって応答処理時間をｔｎＲ＋ｔｎ（ＰＡ）－ＮＲ＋（ｔｎ－１）Ｓだけ短縮することができる。

【００５８】また、図７に示すように、実際の送信では、ＮＩＣ３に対して送信に必要な情報をセットする前処理、ＮＩＣ３が行う送信処理と、送信完了後行う後処理が必要である。特に後処理は、送信完了時に割込みとして起動されるのでレジスタの退避等で数１００ミリ秒が必要となり、従来の方法では送信の度にこの割込みが発生する。本実施の形態による方法では時間ｔ内に発生する送信要求および、受信したデータの応答パケットの送信を、ＮＩＣ３に対し同時に起動をかけるため送信における前処理と後処理は常に１回となる。

【００５９】送信パターン２の一定時間ｔにおける、従来の組込み方法の処理時間ＹＡ、と本実施の形態による組込み方法の処理時間ＹＢとの差Ｓｅｎを求めると、

$$YA=t\times Y\times S \cdots\cdots\cdots\cdots (11)$$
$$YB=S \cdots\cdots\cdots\cdots\cdots (12)$$

式（１１）（１２）より

$$Sen=YA-YB$$
$$=(tY-1)S \cdots\cdots\cdots\cdots (13)$$

Ｙは単位時間内に発生する送信要求数、ＳはＮＩＣ３への送信要求処理時間と送信完了割り込み処理時間の和である。

【００６０】一定時間ｔ内にパケットを受信している場合は、その応答パケットと一緒に送信処理を行うため、ＹＢは０となり、Ｓｅｎ＝ｔＹＳとなる。従って、本実施の形態による組込み方法により、送信時間を（ｔＹ－１）Ｓまたは、ｔＹＳだけ短縮することができる。

【００６１】本送信処理で送信するパケットのタイムアウト処理も送信パターン１と同様にタイマ割り込み中で行う。従来の送信したパケットに対するターンアラウンド時間のタイムアウト処理は、送信したすべてのパケッ

トそれぞれに対しタイマをカウントする必要があるが、本実施の形態による方法のように一定数のパケットを同時に送信することにより、パケットのタイマ管理をグループ化し処理時間を短縮することができる。

【００６２】また、従来はコネクションの接続している間はＴＣＰタスクは起動したままとなり、タイムスライス処理で他のタスクを実行しており、タイムスライス処理に常に一定量のＣＰＵ資源を使用しているので、プロトコル処理や、プリンタファームウェアの処理の動作する効率が非常に悪くなる。

【００６３】本実施の形態による組込み方法では、ジョブのスケジューリングにプライオリティを用いた割り込みを使用し、タイムスライスによるラウンドロビン方式は採用していない。

【００６４】第１の実施の形態によれば、ＴＣＰ／ＩＰ部は受信バッファに受信してあるすべてのパケットを割り込み処理時間ｔ内に処理するので、ＯＳのオーバヘッドを削減することができ、ＣＰＵを他の有効な処理に割り当てることができる。

【００６５】第２の実施の形態

一般に割込み内での動作が多いと、他の処理への影響が大きくなるので、第２の実施の形態では極力割込み内の処理を減らし、大部分の処理をＩＦＣタスクに組み込むようにしたものである。

【００６６】図５は第２の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図であり、受信割込み手段２０とインターバルタイマ２１とインターフェース制御手段としてのＩＦＣタスク２２とから成る。

【００６７】受信割込み手段２０は、パケットの受信時に起動して受信パケットをメモリ５の受信バッファ５ａに格納する毎にＩＦＣタスク２２にメッセージ(message-X)を出力する。インターバルタイマ２１はカウンタ２１ｂがカウントアップする毎にＩＦＣタスク２２にメッセージ(message-S) を出力する。

【００６８】ＩＦＣタスク２２は、メッセージを受信して呼び出し処理を行うメッセージ処理手段２４と、データリンク層（２層）、ネットワーク層（３層）、トランスポート層（４層）の各プロトコルを処理して所定量に達するまでメモリ５のパケット一時バッファ５ｂに蓄積する第１のパケット処理手段２３と、パケット一時バッファ５ｂからパケットを取り出してセッション層（５層）、プレゼンテーション層（６層）、アプリケーション（７層）の各プロトコルを処理し、データをメモリ５のデータバッファ５ｄに出力する第２のパケット処理手段３０と、第１のパケット処理手段２３から呼び出されてメッセージ(message-A) をメッセージ処理手段２４に出力し、メッセージ処理手段２４からの呼び出し応じて第２のパケット処理手段３０に起動を駆けるソケット処理手段２８と、セントロニクス、ＲＳ－２３２Ｃ等のケーブルを制御するＩＦＣタスク内の制御モジュール３２とを備えている。

【００６９】第１のパケット処理手段２３は、受信割込み手段２０からメッセージ処理手段２４にメッセージ(message-X) が出力されることによって起動され、受信バッファ５ａから受信パケットを取り出してデータリンク層（２層）のプロトコル処理を行うネットワークドライバ２５と、ネットワークドライバ２５が処理した受信パケットのネットワーク層（３層）とトランスポート層（４層）の各プロトコルを処理してメモリ５のパケット一時バッファ５ｂに蓄積するプロトコルスタック(TCP/IP 、SPX/IPX 、その他) ２６と、インターバルタイマ２１からメッセージ処理手段２４にメッセージ(message-S) が出力されることによって起動され、コネクションを管理するタイマ処理手段３１とを備えている。

【００７０】第２のパケット処理手段３０は、プロトコルスタック２６を使用することにより実際に他のノードと通信するアプリケーション(FTP、RPRINTER、その他) ２７と、アプリケーション２７が処理したデータのキュー管理を行うジョブ制御モジュール２９とを備えている。

【００７１】ソケット処理手段２８はアプリケーション２７とプロトコルスタック２６の間の通信行う上で共通のインタフェースを提供し、プロトコルスタック２６に呼び出され、アプリケーション２７を指定するメッセージ(message-A) をメッセージ処理手段２４に出力するソケット(Low) ２８ａと、メッセージ処理手段２４に呼び出されメッセージに含まれる内容に基づいてアプリケーション２７を選択するソケット(High)２８ｂとを有する。

【００７２】図６はアプリケーションＦＴＰの構成を示すブロック図である。アプリケーションＦＴＰ２７ａは、イニシャル処理部３５と、メイン処理部３６とから構成される。

【００７３】イニシャル処理部３５は、プリンタ１からのイニシャル要求に対し、ＦＴＰ内のワーキング等の初期化の他、アプリケーション登録をソケット(High)２８ｂに対して行う。

【００７４】メイン処理３６は、メッセージ(message-A) の解析部３７と、下位層の状態をmessage-A によって通知された時に呼び出されるコネクション管理部３８と、受信したパケットの処理を行う際に参照するデータ解析部３９と、データ解析部３９でコマンドと判断されたときに呼ばれるコマンド分岐部４０と、各コマンドの処理群４１とから成る。

【００７５】図７はコネクション管理テーブルの説明図であり、コネクション別に「コネクションの状態」、「処理中のコマンド」、「コネクションＮｏ」、「送信パケット数」、「相手先ポートＮｏ」が配列されてある。コネクション管理テーブルはコネクション管理部３

８により管理され、受信したパケットの処理を行う際ににメッセージ解析部３７より参照される。

【００７６】図８はコネクションの状態と設定値との関係を示す説明図、図９はプロトコルスタックの各処理手段の処理ステップ数と負荷との関係を示す負荷算出テーブル、図１０はパケットを再送する際の処理を示すフローチャートである。

【００７７】次に動作について説明する。クライアントよりランダムに送られてくるパケットは、ネットワーク接続用Ｈ／WであるＮＩＣ３からの受信割り込み信号がＣＰＵ１に入力され、受信割り込みプログラムからなる受信割込み手段２０によってリアルタイムに受信され、受信バッファ５ａに格納される。１パケットを格納する毎に受信割込み手段２０は、図５に矢印（ａ）で示すように、メッセージ(message-X) をＩＦＣタスク２２に対し出力する。

【００７８】ＩＦＣタスク２２のメッセージ処理手段２４は、アイドル時はＩＦＣタスク宛のメッセージのセンスをしている。メッセージ(message-X) よりも先にＩＦＣタスク宛に送られたメッセージがあればＩＦＣタスク２２はその処理を先に実行する。

【００７９】メッセージ処理手段２４はメッセージ(message-X) を受信すると、図５に矢印（ｂ）で示すように、ネットワークドライバ２５を呼び出す。ネットワークドライバ２５は、通常のデータリンク層（２層）の処理を行うが、複数のＮＩＣ３および複数の２層プロトコル(例えばEthernet、IEEE802.3 、IEEE802.5 等）にも対応できる。

【００８０】ネットワークドライバ２５は、データリンク層のプロトコル処理を終了すると、図５に矢印（ｃ）で示すように、上位層を識別して対応するプロトコルスタック２６、例えばＴＣＰ／ＩＰ２６ａを呼び出す。ＴＣＰ／ＩＰ２６ａはＴＣＰとＩＰとを別タスクとせず一つのタスクとして作成されているので、ＴＣＰ－ＩＰ間のオーバーヘッドを削減する。

【００８１】ＴＣＰ／ＩＰ２６ａは、ネットワークドライバ２５から受信したパケットのネットワーク層（３層）とトランスポート層（４層）のプロトコル処理を行い、そのパケットをメモリ５のパケット一時バッファ５ｂに格納し、ネットワークドライバ２５に対して制御を返す。

【００８２】ネットワークドライバ２５は制御をＩＦＣタスク２２から他のタスクへ返し、次のパケットの受信を待つ。

【００８３】以上の処理を数回繰り返すと、パケット一時バッファ５ｂにパケットが一定量蓄積される。ＴＣＰ／ＩＰ２６ａはパケットが一定量に達すると、図５に矢印（ｄ）で示すように、ソケット(Low) ２８ａに対し、対応するアプリケーション２７を指定する情報を出力してパケットの受け取りを要求する。

【００８４】ソケット(Low) ２８ａは、直接、ソケット(High)２８ｂを呼び出さず、図５に矢印（ｅ）で示すように、メッセージ処理手段２４に対しメッセージ(message-A) を出力する。本処理により、メッセージ処理手段２４に起動を駆けられ、セントロニクス、ＲＳ－２３２Ｃ等のケーブルを制御するＩＦＣタスク内の制御モジュール３２が動作するきっかけを与えることができる。

【００８５】メッセージ処理手段２４はメッセージ(message-A) を受信すると、図５に矢印（ｆ）で示すように、ソケット(High)２８ｂを呼び出す。ソケット(High)２８ｂは、図５に矢印（ｈ）で示すように、メッセージ(message-A) の内容に基づいてアプリケーション、例えばＦＴＰ２７ａを呼び出す。

【００８６】ＦＴＰ２７ａは、パケット一時バッファ５ｂに蓄積されたパケットを全て処理し、図５に矢印（ｉ）で示すように、ジョブ制御モジュール２９を呼び出す。

【００８７】ジョブ制御モジュール２９は、図５に矢印（ｊ）で示すように、パケットのデータをメモリ５のデータバッファ５ｄに格納する。

【００８８】ここで、ＦＴＰの動作を図６～図１１を参照して詳細に説明する。ＦＴＰ２７ａはソケット(High)２８ｂから受信したメッセージ(message-A) により呼ばれる関数により起動される。メッセージ解析部３７はメッセージ(message-A) の内容を解析し、コネクションに関する状態を通知するものであれば、コネクション管理部３８を呼び出す。コネクション管理部３８は、メッセージ(message-A) の内容により、図７、８に示すコネクション管理テーブルの「コネクションの状態」の設定値を変更する。

【００８９】メッセージ解析部３７は、メッセージ(message-A) の内容がパケットの受信であれば、データ解析部３９を呼び出す。データ解析部３９は、ソケット(High)２８ｂに対してパケットの獲得を要求し、受信したパケットの「コネクションの状態」を知るために、図７に示したコネクション管理テーブルの「コネクションの状態」を参照する。

【００９０】「コネクションの状態」が、図８に示したように設定値４であれば、獲得したパケットは、印字データと判断し、ジョブ制御モジュール２９にパケットを渡す。「コネクションの状態」が、設定値０、又は３である場合には、パケットを受信する状態ではないので、獲得したパケットを捨てる。

【００９１】「コネクションの状態」が、設定値１、又は２である場合には、獲得したパケットは、コマンドであると判断してコマンド分岐部４０を呼び出す。コマンド分岐部４０は、「コネクションの状態」が、設定値１で受信したパケットがログイン要求（ＵＳＥＲコマンド）である場合には、ログイン処理と判断し、コネクション管理部３８を呼び出す。

【００９２】「コネクションの状態」が、設定値２である場合には、ログイン済であるので、ログイン要求以外のコマンドを受け付けられるので、該当するコマンド処理部４１を呼び出す。

【００９３】コマンド処理部４１は送受信を必要とするコマンドを受け付けた場合には、クライアントに対して処理し、データのコネクションの確立を行うために、下位層に対してコネクションの確立要求を出す。

【００９４】コネクションが確立されるまでの間は、他の処理に制御を渡すため、データコネクションの確立後の処理をコネクション管理テーブルの「処理中のコマンド」に書き込み、保持する。

【００９５】このように、各処理が独立してコネクション管理テーブルの「コネクションの状態」エリアの設定値によって動作するので、複数のコネクション管理を行うことができる。

【００９６】パケット送信時のタイマ処理は、図５に矢印（１）～（ｎ）で示すように行なわれる。まず、一定時間毎にインターバルタイマ２１は、矢印（１）で示すように、ＩＦＣタスク２２のメッセージ処理手段２４に対してメッセージ(message-S) を出力する。

【００９７】メッセージ処理手段２４はメッセージ(message-S) を受信すると、矢印（ｍ）に示すように、タイマ処理手段３１を呼び出す。タイマ処理手段３１は、矢印（ｎ）に示すように、必要に応じてプロトコルスタック２６を呼び出す。

【００９８】タイマ処理手段３１及びタイマ処理手段３１によって呼び出されるプロトコルスタックの動作を図１０に基づいて説明する。プロトコルスタック２６は、例えばＴＣＰ／ＩＰ２６ａが呼び出されているものとする。まず、ステップS1 とステップS2 とは、全コネクションにおいて毎回実行しなければならない処理で、タイマ処理手段３１を起動することにより必ず実行される。

【００９９】即ち、「コネクションの状態」のチェック、応答待ちパケットのタイマカウント処理、パケットの再送処理、Ｍａｘ　Ｔｉｍｅｏｕｔ処理等を実行する。

【０１００】その他のステップは、プロトコル上必ず毎回実行する必要のない処理で、処理時間のかかるものが多く、時間的に処理量のばらつきが大きい。これらの処理は、図９に示すような処理ステップ数と対応する負荷係数Ｇを用いる。

【０１０１】まずステップS3 で、負荷変数を初期化する。ステップS4 で、コネクション単位に要求される処理を行い、１コネクションの処理が終了するまで行う。

【０１０２】ステップS5 で１コネクションの処理の最後に行った処理の負荷係数Ｇの合計を負荷変数に加算する。ステップS6 で、負荷変数が一定値(MAX) 以上であれば、ステップS8 で処理を中断してタイマ関数を抜ける。このとき次にタイマ関数が呼び出された時に処理を開始するコネクションの位置(コネクションNo.)をメモリ５のグローバルエリア内の変数に記憶しておく。

【０１０３】負荷変数が一定値(MAX) 以下であれば、全コネクションの処理を終了するまでステップS3 ～ステップS7 を繰り返す。

【０１０４】負荷変数が一定値(MAX) 以下のまま、すべてのコネクションの処理を終了した場合はタイマ処理を抜ける。

【０１０５】第２の実施の形態によれば、ネットワークプロトコル(２層～７層) 処理をシングルタスク上でソケットインタフェースにより下位層と上位層とに分け、ＯＳの基本機能であるメッセージによって各層の処理手段に起動をかけ得るようにしたので、シングルタスク上で２つの処理に優先順位をつけることができ、また、ネットワークプロトコル処理中に割込みにより他のインターフェース処理を受け付けることができるので、同一タスク上で他のホストインタフェース(セントロニクス、RS-232C)との共存を可能とした。

【０１０６】また、タイマ処理手段を設けたことにより、一時期に集中しがちな処理の負荷を分散し平均化させ、安定したシステム状態を保つことができる。このことにより、プリンタではオーバーランや、オーバーフロー等の性能を向上させ、ネットワーク処理の集中に関係なく安定した性能を保つことができる。

【０１０７】また、シングルタスク上で動作するアプリケーションでマルチコネクションを確立し、管理することが可能となる。

【０１０８】第３の実施の形態

第３の実施の形態は受信バッファ５ａの使用量を定期的に検知して許容コネクション数を変化させ、処理飽和状態からのリカバリーを迅速にするとともに、低要求コネクション確立時にも、効率良く動作させるようにしたものである。

【０１０９】図１１は第３の実施の形態によるインターバルタイマ処理の要部を示すブロック図、図１２は許容コネクションを増減させる際に使用するテーブルである。

【０１１０】まず、図５に示したタイマ処理手段３１に受信バッファ５ａの使用量をセンスするバッファ数センス部５０と、図１２に示したテーブル５１ａを参照して処理負荷を判断する判断部５１と、許容コネクション数を設定するコネクション数設定部５２とを設ける。

【０１１１】次に動作について図１３に従って説明する。図１３は第３の実施の形態の動作を示すフローチャートである。ステップS1 でバッファ数センス部５０はインターバルタイマ２１の周期で受信バッファ５ａの使用量をセンスし、ステップS2で受信バッファ５ａの使用量をグローバルに確保したメモリ５の記憶領域Ｘに格納する。また、１回目のセンス時は、「前回のセンス時

の受信バッファ使用量（Y）」データは存在しないのでc、メモリ5の記憶領域Yを0xFFFFにしておく。

【0112】ステップS3 で記憶領域Yの内容が0xFFFFか否かをチェックし、0xFFFFの場合にはステップS5 に分岐し、否の場合はステップS4 に分岐する。ステップS4 で「前回のセンス時の受信バッファ使用量（Y）」データが存在するので、「今回のセンス時の受信バッファ使用量（X）」を用い、図12に示すテーブル51aとの比較により、許容コネクション数の変更量を決定し、メモリ5の記憶領域Zへ書き込む。

【0113】ステップS5 で、次のセンスに備え、「今回センス時の受信バッファ使用量（X）」を「前回のセンス時の受信バッファ使用量（Y）」へコピーする。

【0114】ステップS6 で、記憶領域Zの内容が0か否かをチェックして、0ならばステップS1 に戻って次のインターバルタイマ21による起動を待ち、否ならばステップS7 で記憶領域Zの内容に応じたコネクション数を増減させる。コネクション数設定部52は、記憶領域Zの内容が0以外の時に起動し、記憶領域Zの内容に応じたコネクション数を増減させる。本処理は、記憶領域Zの内容が正の数である時には、各コネクションに必要な領域の確保とイニシャライズ行う。記憶領域Zの内容が負の数である時には各コネクションに必要な領域を使用不可能にする。許容数までコネクションが接続している場合は、コネクションが切断され次第各コネクションに必要な領域を使用不可能にする。

【0115】図12を用いてコネクション数設定部の動作を説明する。コネクション数設定部52は、記憶領域Zの内容が0以外の時に起動し、記憶領域Zの内容に応じたコネクション数を増減させる。記憶領域Zの内容が正の数である時には、各コネクションに必要な領域の確保とイニシャライズ行う。記憶領域Zの内容が負の数である時には各コネクションに必要な領域を使用不可能にする。許容数までコネクションが接続している場合は、コネクションが切断され次第各コネクションに必要な領域を使用不可能にする。

【0116】現在及び前回の使用受信バッファ量が全受信バッファ量の20% 以下であった場合は、要求負荷の低いコネクションが全コネクションの大半を占めている可能性が高いので、コネクションの許容数を1 つ増加させる。

【0117】また、現在の使用受信バッファ量が全受信バッファ量の80% 以上であった場合には、後工程の処理の負荷が高く、受信バッファのオーバフローが予想されるのでコネクションの許容数を1 つ減少させる。

【0118】現在の使用受信バッファ量が全受信バッファ量の21% ～50% の間で、前回の使用受信バッファ量が51% 以上であった場合は、処理負荷が下降していることを示しているので、急激な処理負荷の下降に備えコネクションの許容数を1 つ増加させる。

【0119】現在の使用受信バッファ量が全受信バッファ量の51% ～80% の間で、前回の使用受信バッファ量が50% 未満であった場合は、処理負荷が上昇していることを示しているので、急激な処理負荷の上昇に備えコネクションの許容数を1 つ減少させる。

【0120】第3の実施の形態によれば、現在行っている処理の重さにより確立するコネクション数を変化させることにより、処理飽和状態からのリカバリーを早くし低要求コネクション確立時にも、より効率良く動作させることができる。

【0121】第4の実施の形態

第4の実施の形態は、図1に示した印刷部10の展開タスク10a等からの高負荷通知に対し、インターバルタイマ21のタイマ間隔を可変とすることにより、展開タスク10a等のプライオリティの低いタスクが処理負荷の高い状態にある時に、IFCタスク22等のプライオリティの高いタスクを制御できるようにしたものである。

【0122】図14は第4の実施の形態による負荷監視処理のブロック図である。負荷監視部60は図5に示した展開タスク10a等の処理の負荷を監視するためにプライオリティの低いタスク内に設けられている。負荷通知エリア61はメモリ5のグローバルエリアに設けられ、負荷監視部60から通知を受信し、高負荷時に他の処理へそれを通知する。タイマ間隔値変更手段21cはインターバルタイマ21に設けられ、高負荷時にインターバルタイマ21のカウンタ21bのタイマ間隔値を変更する。

【0123】図15はプリンタにおける展開処理とエンジン処理等とのタイムチャートであり、（a）～（d）は展開処理、エンジン処理、タイマ処理、その他の処理を示している。

【0124】時間Wは、エンジン処理期間T1 間に展開処理が動作できる時間を示し、この時間内に一定のデータの展開が終了しない場合にオーバランが発生する。タイマ処理と、その他の処理（共に展開処理よりもプライオリティは高い）が時刻t1、t2 、t4 、t5 で発生すると、実際の展開処理動作は、矢印で示す範囲となり、オーバランが発生しやすくなる。

【0125】図16は第4の実施の形態の動作を示すフローチャートである。

【0126】次に動作について図16のステップに従って説明する。ステップS1 でインターバルタイマ21のタイマ処理21aは、タイマ割込み毎にカウントするカウンタ21bがタイマ間隔値T以上なったかをセンスし、以下だった場合はステップS2 でカウンタ21bをインクリメントし、以上だった場合はステップS3 でプロトコルスタック内のタイマ処理手段31を呼出すためにIFCタスク22に対してメッセージ(message-S) を出力する。ステップS4 でカウンタ21bは0にリセッ

トされる。

【０１２７】ステップS5 でタイマ間隔値変更手段２１ｃは負荷通知エリア６１を参照し、負荷監視部６０からの高負荷通知が受信されていると、ステップS8 でカウンタ２１ｂのタイマ間隔値Ｔを２倍にのばす。高負荷通知が受信されていない場合には、ステップS6 でカウンタ２１ｂのタイマ間隔値Ｔを調べ、変更前のデフォルト値であった場合には本処理を終了する。変更が入っている場合には、ステップS7 でタイマ間隔値Ｔを半分に短縮する。負荷監視部６０から負荷通知エリア６１に高負荷通知が出力され続けていると、タイマ処理手段３１からポストされて起動されるプロトコルスタック２６中のタイマ処理は２倍単位でその間隔が延び、負荷通知エリア６１に高負荷通知が出力されていない時間が続くと、デフォルトのタイマ間隔値まで二分の一単位で間隔は短縮される。

【０１２８】本実施の形態をプリンタに適用し、展開タスク等からの高負荷通知に対し、タイマ処理の間隔を可変とすることにより、図１５の時間Wに示す間隔内のタイマ処理時間を少なくすることが可能となる。また、図１４に示す負荷通知エリアをグローバル領域に設定することにより、タイマ処理以外の高負荷処理へも同時に通知することができ、対策を入れることが可能となる。

【０１２９】第４の実施の形態によれば、展開タスク等からの高負荷通知に対し、タイマ処理の間隔を可変とすることにより、展開タスク等のプライオリティの低いタスクが処理負荷の高い状態にある時に、ＩＦＣ等のプライオリティの高いタスクを制御できる。このことにより、プリンタではオーバーランや、オーバーフロー等の性能を向上させ、ネットワーク処理の集中に関係なく安定した性能を保つことができる。

【０１３０】第５の実施の形態

第１の実施の形態から第４の実施の形態までは単独のプリンタ記述言語を備えたプリンタに対応したが、第５の実施の形態はプリンタが複数のプリンタ記述言語、プリンタコントロールランゲエジ（PRINTER CONTROL LANGUAGE: 以後ＰＣＬと記す）、ポストスクリプト（POST SCRIPT:以後ＰＳと記す）、プリンタジョブランゲエジ（PRINTER JOB LANGUAGE: 以後ＰＪＬと記す）等を備えた場合に対応できるようにしたものである。

【０１３１】図１７は第５の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図であり、第２の実施の形態によるプリンタと異なるところは、ＰＣＬ、ＰＳ、ＰＪＬ等のプリンタ記述言語に対応するＩＦＣタスク４０から共通に使用できるようにネットワークドライバ２５を再入可能（一つの関数からコールされ、リターンする前に他の関数からコールすることが可能である仕様）にした点と、セントロニクス、ＲＳ－２３２Ｃ等のインターフェースをネットワークドライバ２５と同様に共通関数にした点である。

【０１３２】プリンタ言語判別部４１はプリンタ記述言語に対応する各ＩＦＣタスク４０やセントロニクス、ＲＳ－２３２Ｃ等のインターフェースと、各プリンタ記述言語に対応する編集部、例えばＰＣＬ用編集部４２、ＰＳ用編集部４３に接続され、各ＩＦＣタスク４０から出力されるデータのプリンタ記述言語を判別して対応する編集部に出力している。

【０１３３】図１８はＩＦＣタスクの構造を示すブロック図であり、第２の実施の形態によるＩＦＣタスクの構造と異なるところは、ネットワークドライバ２５がＩＦＣタスク４０の外にある点と、上位層から呼ばれる共通化ドライバ関数４４をＩＦＣタスク４０の内部に設けた点である。

【０１３４】図１９は共通関数部の構造を示すブロック図であり、ＩＦＣタスク４０のアクセスを制御するドライバ（Ethernet方式、IEEE802.2 方式、IEEE802.3 方式、SNAP方式等）をまとめたネットワークドライバ２５と、ネットワーク接続用Ｈ／Ｗを制御するドライバをまとめたＮＩＣドライバ４５とからなる。

【０１３５】次に動作を図１８を参照して説明する。第２の実施の形態と異なるところは、メッセージ処理手段２４がＮＩＣドライバ４５からなる受信割り込み手段２０から矢印（ａ）に示すメッセージを受信すると、ＩＦＣタスク４０の外にあるネットワークドライバ２５を関数コールする点であり、その他の動作は第２の実施の形態と同様である。

【０１３６】第５の実施の形態によれば、メモリ容量の大きいネットワークドライバをＰＣＬ、ＰＳ、ＰＪＬ等のプリンタ記述言語に対応するＩＦＣタスクに設けることなく、各ＩＦＣタスクから共通に使用できるようにしたので、メモリを効率よく使用できる。

【０１３７】第６の実施の形態

第６の実施の形態によるプリンタのネットワークアーキテクチャは、第５の実施の形態と同じである。メッセージ処理手段２４は各処理手段からのメッセージを受信するイベントコントロールブロック（Event Control Block:以後ＥＣＢと記す）からメッセージを取り出して処理するが、ＥＣＢが複数あって、処理する優先度をＥＣＢ間に設けてあればタスク内の処理にも優先度を設けることができ、効率良くＣＰＵ１を動作させることができるが、ＥＣＢが一つの場合には、タスク内の処理に優先度を設けることができないのでＣＰＵ１を効率良く動作できなくなる。

【０１３８】第６の実施の形態はＥＣＢが一つの場合にも効率の良い動作ができるようにしたものである。

【０１３９】図２０は第６の実施の形態によるメッセージ処理手段の動作を示すブロック図である。メッセージ処理手段２４はＥＣＢ４６とソケット処理手段２８の上位側待ち行列ＷＡＰＦQueue ４７とを参照し、ＥＣＢ４６からメッセージを取り出してプロトコルスタック２６

へプロトコルを処理するように要求するか、上位側待ち行列ＷＡＰＦQueue ４７の処理をコールするかを判定する。

【０１４０】図２１は図２０に示したメッセージ処理手段の動作を示すフローチャートである。メッセージ処理手段２４はステップS1 でソケット処理手段２８に上位層待ち行列ＷＡＰＦQueue ４７が生成されてあるか否かをチェックして生成されてあるならばステップS2 に分岐し、否ならばステップS4 に分岐する。

【０１４１】ステップS2 で上位層待ち行列数が所定量、例えば１０個未満か否かをチェックして１０個未満ならばステップS3 に分岐し、否ならばステップS5 に分岐する。ステップS3 でＥＣＢ４６にメッセージが受信してあるか否かをチェックして受信してあるならばステップS4 に分岐し、否ならばステップS5 に分岐する。ステップS4 でプロトコルスタック２６へプロトコルを処理するように要求し、ステップS5 でソケット（Ｈｉｇｈ）２８ｂをコールする。

【０１４２】尚、本実施の形態は第２の実施の形態にも適用できる。

【０１４３】第６の実施の形態によれば、ＯＳからポストされたメッセージを格納するＥＣＢを一つのタスクに複数設けられない場合にも、タスク内の処理に優先度を設けることができので、ＣＰＵを効率良く動作させることができる。

【０１４４】第７の実施の形態

第７の実施の形態によるプリンタのネットワークアーキテクチャは、第５の実施の形態と同じである。ジョブ制御モジュール２９はデータのクライアント識別子を格納する共通テーブルを有し、データを出力する際にクライアント識別子を格納し、上位層からデータ送信要求を受ける際には共通テーブルに基づいてクライアント識別子を解析し、同一コネクションを通じてパケットを送受信するようにしたものである。

【０１４５】図２２は第７の実施の形態を示すブロック図である。２のパケット処理手段３０はジョブ制御モジュール２９にデータのクライアント識別子（以後クライアントＩＤと記す）等を登録する共通テーブルを有する。

【０１４６】共通テーブルにはデータを印刷部の編集タスクに送信する際に使用するＤＭＤ４８と、プリンタ言語判別部４１からデータ送信要求を受ける際に登録するSendDescriptor ４９とがある。例えば、ネットワークドライバ２５を経由して３つのパケットを受信したとする。各パケットはプロトコルを処理されてアプリケーションＡＰ1 、ＡＰ2 、ＡＰ4 を通じてジョブ制御モジュール２９に送信されてくるものとする。

【０１４７】クライアントＩＤ　C1 、C2 、C3 は受信割り込み手段２０がパケットを受信した際にＤＭＤ４８に登録される。ジョブ制御モジュール２９はアプリケーションＡＰ1 、ＡＰ2 、ＡＰ4 から受信する印刷データとともにクライアントＩＤC1 、C2 、C3 をプリンタ言語判別部４１に送信する。クライアントＩＤは３２ビットで構成される。最初の８ビットはＩＦＣタスク４０、セントロニクス、ＲＳ－２３２Ｃ等のインターフェースの種別を示し、次の２４ビットは各インターフェースによって固有となる。例えばＩＦＣタスク４０の場合には、８ビットがアプリケーションを示すＩＤ（ＡＰ1 、ＡＰ2 、・・・・）で、残りの１６ビットが各アプリケーションにおける通番となる。このフォーマットにより同一のアプリケーションにクライアントの接続が集中しても対応することができる。

【０１４８】プリンタ言語判別部４１は受信したクライアントに対してデータを送信する場合にはインターフェースの種別をクライアントＩＤの最初の８ビットから判断し、そのインターフェースのSend Descriptor ４９に登録する。

【０１４９】ジョブ制御モジュール２９はSend Descriptor ４９を参照してアプリケーションを識別するビットを読み取り、対応するアプリケーションをコールする。

【０１５０】尚、本実施の形態は第２の実施の形態にも適用できる。

【０１５１】第７の実施の形態によれば、ＩＦＣタスク等のインターフェースを使用してパケットの授受を行う上位層が、クライアントＩＤの中身を意識することなく応答パケットの送信先を特定するので、要求するクライアントに正確に応答パケットを送信できる。

【０１５２】第８の実施の形態

第８の実施の形態によるプリンタのネットワークアーキテクチャは、第５の実施の形態と同じである。第８の実施の形態はプリンタ等に組み込まれた受信バッファのサイズが小さい場合に、効率的な受信バッファ空き容量を算出して受信速度を向上させるようにしたものである。

【０１５３】図２３はプリンタ言語判別部４１とジョブ制御モジュール２９との間のデータ受け渡し処理を示す説明図、図２４は送信側ＴＣＰと受信側ＴＣＰとの間のデータ受け渡し処理を示す説明図、図２５は第８の実施の形態を示すブロック図である。第７の実施の形態で説明したように、プリンタ言語判別部４１は処理状況に応じて受信データをジョブ制御モジュール２９に要求するが、図２３に示すように、データ要求のポーリング間隔が常に一定は限らない。

【０１５４】また、ＴＣＰ間の受信処理では、図２４に示すように、受信パケットを複数受信して一つの応答パケットを返すブロック転送方式を使用している。さらに、トランスポート層の機能を使用するなど、送受信端末間の距離制限は事実上無いに等しい。従って、パケット転送時間Ｔは非常に短い場合もあるが、非常に長い場合もある。

【０１５５】そこでジョブ制御モジュール２９に定期的

にポーリング間隔を監視して、ジョブ制御モジュール２９からプリンタ言語判別部４１に転送される単位時間内のデータ数Ｒを計数する転送データ計数手段５０を設ける。

【０１５６】また、ＴＣＰの受信パケット処理時、パケット転送時間Ｔを算出するパケット転送時間更新手段５１をプロトコルスタック２６に設ける。

【０１５７】また、ジョブ制御モジュール２９には、パケット転送時間Ｔと単位時間にデータを上位層に転送する転送データ数Ｒと現在の受信バッファ空容量Ｗとから次回のパケットブロック転送において受信可能なデータ容量Ｘを算出する受信可能データ容量算出手段５２を設ける。

【０１５８】受信可能なデータ容量Ｘは、

Ｘ＝（Ｒ×Ｔ＋Ｗ）・・・・・・・・・・・・（１４）

で算出される。

【０１５９】メモリ５にはパケット転送時間Ｔを格納する格納エリア５３、ＮＩＣドライバ４５が受信パケットを受信する毎に更新される受信バッファ空容量Ｗを格納する格納エリア５４、受信可能データ容量算出手段５２が算出した受信可能なデータ容量Ｘを格納する格納エリア５５が設けてある。

【０１６０】プロトコルスタック２６の応答パケット作成手段５６は応答パケットを作成する際に格納エリア５５の内容を応答パケットに盛り込んで送信する。

【０１６１】尚、本実施の形態は第２の実施の形態にも適用できる。

【０１６２】第８の実施の形態によれば、実際の受信バッファサイズにとらわれず、後工程の処理速度と前工程の受信速度を接近させることにより、高速で効率的な受信を実現できる。

【０１６３】第９の実施の形態

第９の実施の形態によるプリンタのネットワークアーキテクチャは、第５の実施の形態と同じである。第９の実施の形態は、受信バッファが受信データで満杯になり、送信側からの接続を確認するポーリングパケットが受信できず、そのパケットに対する応答パケットを送信できない状態が発生する。その状態が長時間続いた場合にも接続が切断されないようにしたものである。

【０１６４】図２６は第９の実施の形態を示すブロック図であり、図１９に示したデータリンク層以下を処理するＮＩＣドライバの受信構造を示す。ＮＩＣドライバ４５は、第７の実施の形態で説明した通り、受信パケットを自層及びその上位層で共通管理する共通テーブルにリンクされている。共通テーブルは各層が処理した情報や自層の後工程である層に対して通知する情報を格納する領域としてメモリ５に設けてある。

【０１６５】受信バッファ５ａには常時空けておく「常時空きエリア」と、常時使用する「常時使用エリア」とを設けておく。ＮＩＣドライバ４５は「常時空きエリア」と「常時使用エリア」とを認識し、受信パケットを優先的に「常時使用エリア」に受信するとともに共通テーブルに情報（受信データサイズ、受信データ格納領域の先頭アドレス、等）を格納する。「常時使用エリア」が受信パケットで満たされると、「常時空きエリア」に受信できるように制御する。

【０１６６】ＮＩＣドライバ４５は「常時空きエリア」に受信している状態でも、受信パケットの処理が進み、「常時使用エリア」に空き領域ができた場合には、以降「常時使用エリア」に受信するように制御する。「常時空きエリア」は１～２パケット分の領域からなり、このエリアにパケットを受信した場合には「常時空きエリア」にリンクした共通テーブル（特）にその情報を格納する。

【０１６７】図２７は第９の実施の形態の動作を示すフローチャートであり、（Ａ）はＮＩＣドライバの動作を示し、（Ｂ）はＴＣＰバイパス処理の動作を示す。

【０１６８】先ず、ＮＩＣドライバ４５の動作を説明する。ステップS1 で「常時空きエリア」にパケットが受信されたか否かをチェックし、受信ならばステップS2 に分岐し、否ならばステップS5 に分岐する。ステップS2 で共通テーブル（特）に空きがあるか否かをチェックし、空きがあるならばステップS3 に分岐し、否ならばステップS6 に分岐する。ステップS3 で「常時空きエリア」に受信されたパケットの情報を共通テーブル（特）に格納する。

【０１６９】ステップS4 でＴＣＰバイパス処理を呼び出し、ＴＣＰバイパス処理後、ステップS1 に戻る。ステップS5 では通常の処理を行ってステップS1 に戻る。ステップS6 では「常時空きエリア」に受信したパケットを破棄してステップS1に戻る。

【０１７０】ステップS7 で受信パケットがパケット送信元からのポーリングパケットか否かをチェックし、ポーリングパケットならばステップS8 に分岐し、否ならばステップS9 に分岐する。受信側のＴＣＰは受信バッファが満杯になった等の理由で一時的に受信を止めたい場合には、応答パケットの所定のエリアに０を格納して送信側のＴＣＰに送信する。送信側のＴＣＰはこの応答パケットを受信すると、パケットの送信を停止する。そして、パケットの送信を開始するタイミングを獲得するために、受信側のＴＣＰにポーリングパケットを定期的に送信する。

【０１７１】ステップS8 で応答パケットを作成して送信する。ステップS9 で「常時空きエリア」に受信したパケットを破棄する。

【０１７２】尚、本実施の形態は第２の実施の形態にも適用できる。

【０１７３】第９の実施の形態によれば、受信バッファが受信データで満杯になり、長時間受信バッファが空かない場合でも、送信側からの接続を確認するポーリング

パケットが受信でき、そのパケットに対する応答パケットを送信できるので、コネクションを維持できるとともに、その間低負荷の処理により接続を保つことができる。

【０１７４】

【発明の効果】本発明は、以上説明したように構成されているので以下に記載される効果を奏する。ネットワーク層、トランスポート層を受信パケット一括処理手段という一つの機能で処理するようにしたので、プロトコル処理を１回の起動で、受信バッファに受信された全パケットを処理できるので、ＯＳのオーバヘッド量を減少させることができる。

【０１７５】また、ネットワーク層、トランスポート層を一括処理することにより、プロトコルをネットワーク層、トランスポート層別に処理する場合に比べてメモリ上の作業領域を削減できる。

【０１７６】さらに、それぞれのコネクションに属する最後の受信パケットに対する応答パケットを割込み中にネットワークに一括して送信する送信パケット一括処理手段を備えたことにより、送信処理時間を大幅に短縮でき、その分他の処理にＣＰＵを動作させることができるので、効率良く動作するネットワークプロトコルのインプリメント手法を提供できる。

【０１７７】同様に、少なくともネットワーク層、トランスポート層を第１のパケット処理手段という一つの機能で処理するようにし、セッション層、プレゼンテーション層、アプリケーション層を第２のパケット処理手段という一つの機能で処理するようにして、その２つの機能間にソケット処理手段及びメッセージ処理手段を介在させて処理の優先度を持たせるようにしたことにより、ＯＳのオーバヘッド量を減少させることができる。

【０１７８】また、ネットワーク層、トランスポート層を一括処理することにより、プロトコルをネットワーク層、トランスポート層別に処理する場合に比べてメモリ上の作業領域を削減できる。

【０１７９】また、低機能ＯＳを用いて他のインターフェース処理をも受け付けるできるようにしたので、同一タスク上で他のホストインタフェース(セントロニクス、RS-232C)との共存を可能にできる。

【図面の簡単な説明】

【図１】第１の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図である。

【図２】第１の実施の形態によるネットワークの構成図である。

【図３】図１に示したネットワークアーキテクチャのタイムチャートである。

【図４】従来と本発明との送信処理の比較を示すタイムチャートである。

【図５】第２の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図である。

【図６】アプリケーションＦＴＰの構成を示すブロック図である。

【図７】コネクション管理テーブルの説明図である。

【図８】コネクションの状態と設定値との関係を示す説明図である。

【図９】処理ステップ数と負荷との関係を示す負荷算出テーブルである。

【図１０】パケットを再送する際の処理を示すフローチャートである。

【図１１】第３の実施の形態の要部を示すブロック図である。

【図１２】許容コネクションを増減させる際に使用するテーブルである。

【図１３】第３の実施の形態の動作を示すフローチャートである。

【図１４】第４の実施の形態による負荷監視処理のブロック図である。

【図１５】プリンタにおける展開処理とエンジン処理等とのタイムチャートである。

【図１６】第４の実施の形態の動作を示すフローチャートである。

【図１７】第５の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図である。

【図１８】ＩＦＣタスクの構造を示すブロック図である。

【図１９】共通関数部の構造を示すブロック図である。

【図２０】第６の実施の形態を示すブロック図である。

【図２１】メッセージ処理手段の動作を示すフローチャートである。

【図２２】第７の実施の形態を示すブロック図である。

【図２３】プリンタ言語判別部とジョブ制御モジュールとの間のデータ受け渡し処理の説明図である。

【図２４】ＴＣＰ間のデータ受け渡し処理の説明図である。

【図２５】第８の実施の形態を示すブロック図である。

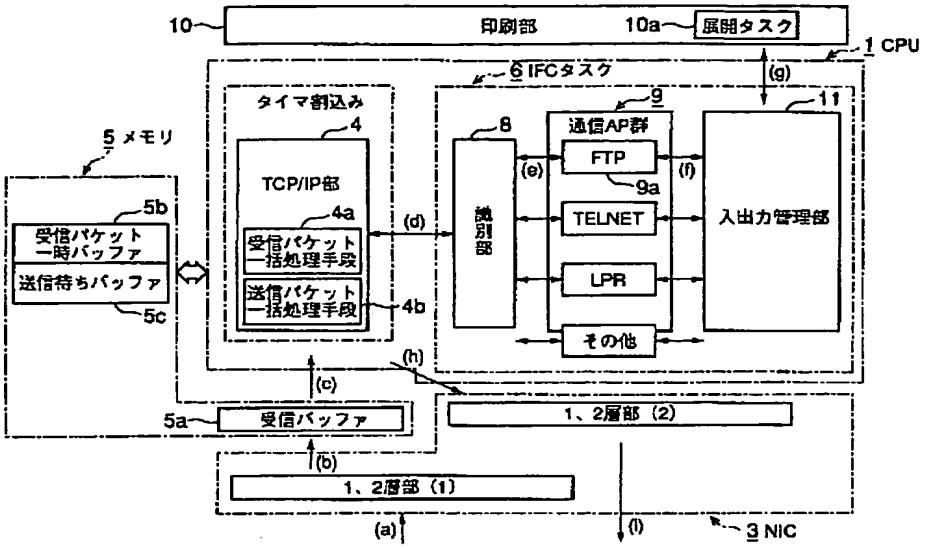
【図２６】第９の実施の形態を示すブロック図である。

【図２７】第９の実施の形態の動作を示すフローチャートである。

【符号の説明】

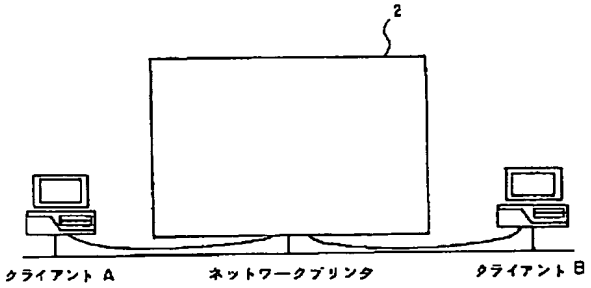
１　ＣＰＵ
３　ＮＩＣ
４　ＴＣＰ／ＩＰ部
５　メモリ
６、２２、４０　ＩＦＣタスク
２３　第１のパケット処理手段
３０　第２のパケット処理手段

【図１】



第1の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図

【図２】



第１の実施の形態によるネットワークの構成図

【図３】

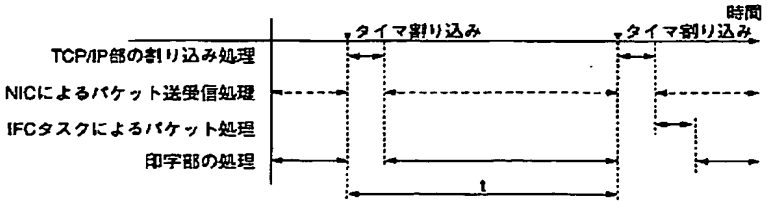


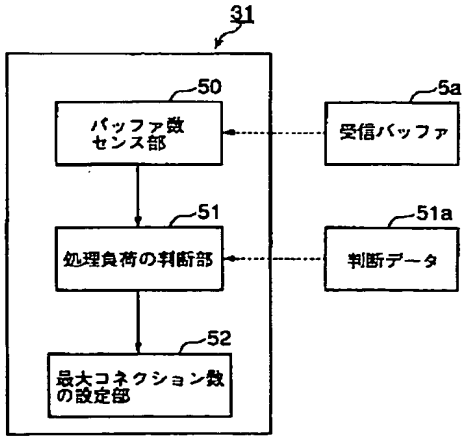
図1に示したネットワークアーキテクチャのタイムチャート

【図７】

コネクション1	コネクション2	コネクション3	コネクションn
コネクションの状態	コネクションの状態	コネクションの状態	コネクションの状態
処理中のコマンド	処理中のコマンド	処理中のコマンド	処理中のコマンド
コネクションNo.	コネクションNo.	コネクションNo.	コネクションNo.
送信パケット数	送信パケット数	送信パケット数	送信パケット数
相手先ポートNo.	相手先ポートNo.	相手先ポートNo.	相手先ポートNo.

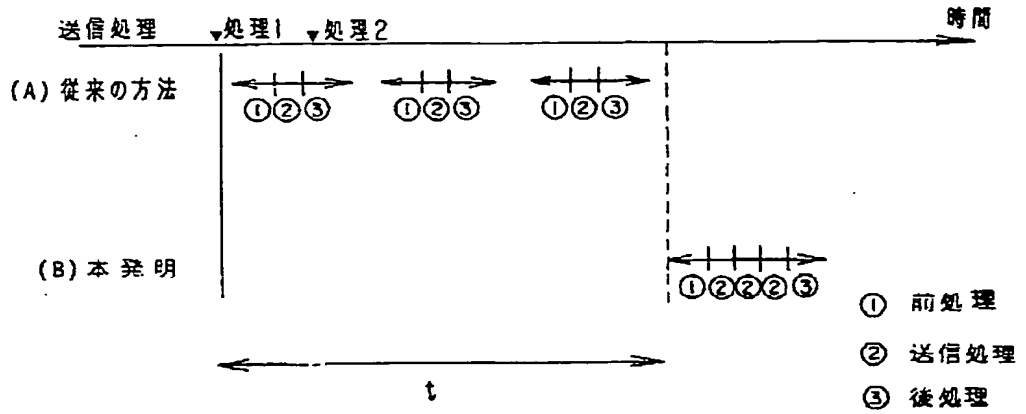
コネクション管理テーブルの説明図

【図１１】



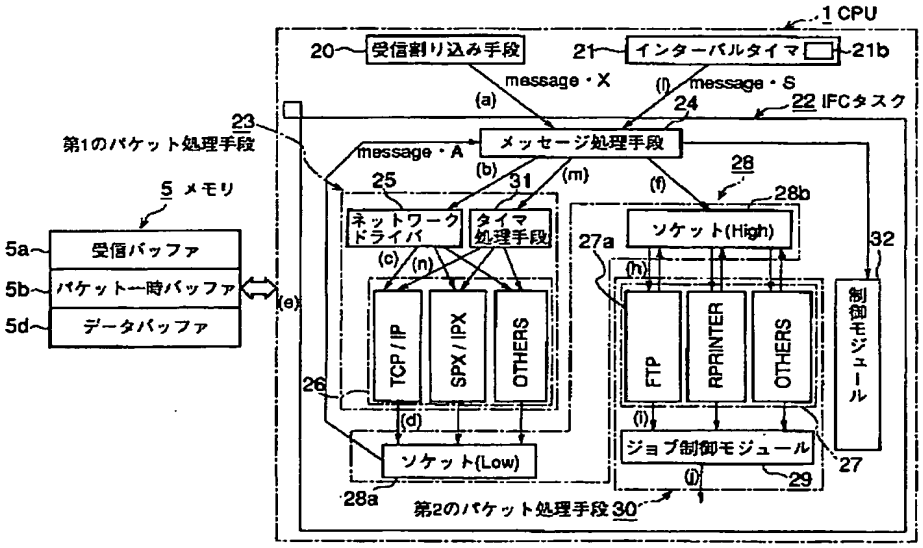
第3の実施の形態の要部を示すブロック図

【図4】



従来と本発明との送信処理の比較を示すタイムチャート

【図5】



第2の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図

【図8】

コネクションの状態	設定値
コネクションが確立していない状態	0
制御コネクションが確立した状態	1
制御コネクションからログインを許可した状態	2
データコネクションを確立要求中の状態	3
データコネクションが確立した状態	4

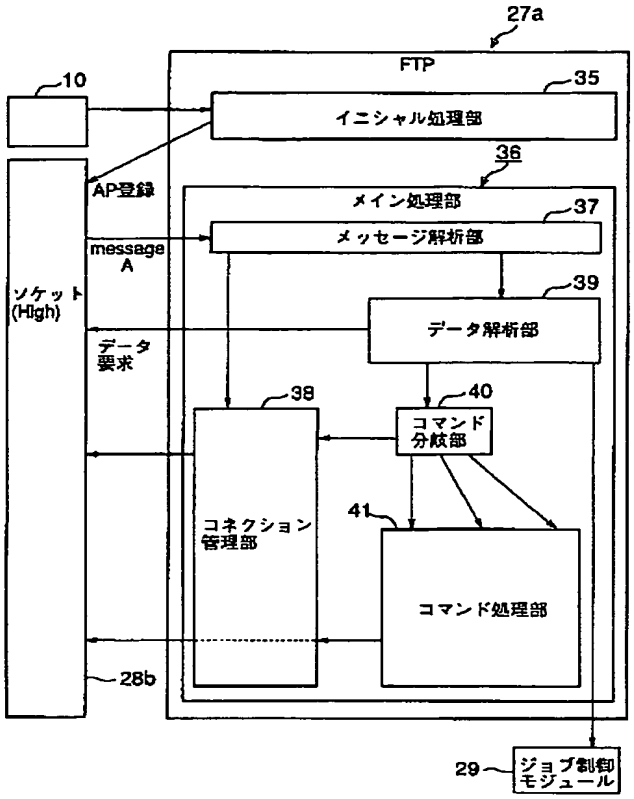
コネクションの状態と設定値との関係を示す説明図

【図9】

	TCP/IP		SPX/IPX		Others	
	STEP数	G	STEP数	G	STEP数	G
処理1	70	70	40	40	なし	0
処理2	120	120	113	113	130	130
処理3	120	120	170	170	178	178
処理4	なし	0	70	70	104	104

処理ステップ数と負荷との関係を示す負荷算出テーブル

【図６】



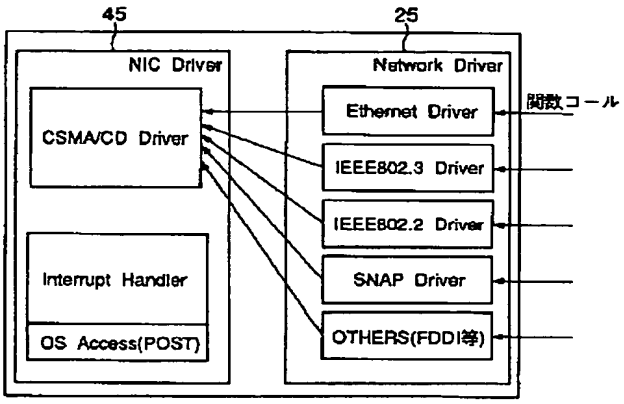
アプリケーションFTPの構成を示すブロック図

【図１２】

現在の使用受信バッファ量	前回センス時の使用受信バッファ量	コネクション数の変更量
２０%以下	２０%以下	１増加
２１～５０%	５１%以上	１増加
５０～８０%	５０%未満	１減少
８０%以上	－	１減少

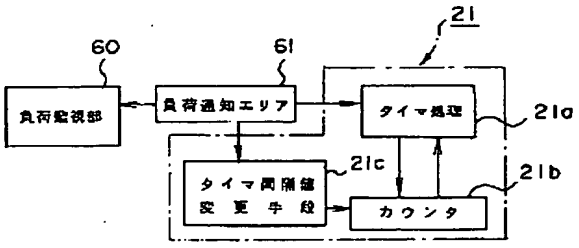
許容コネクションを増減させる際使用するテーブル

【図１９】



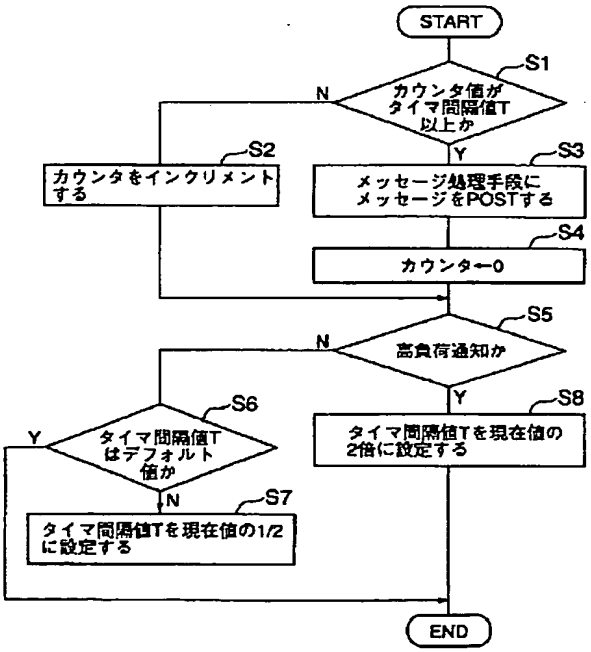
共通関数部の構造を示すブロック図

【図１４】



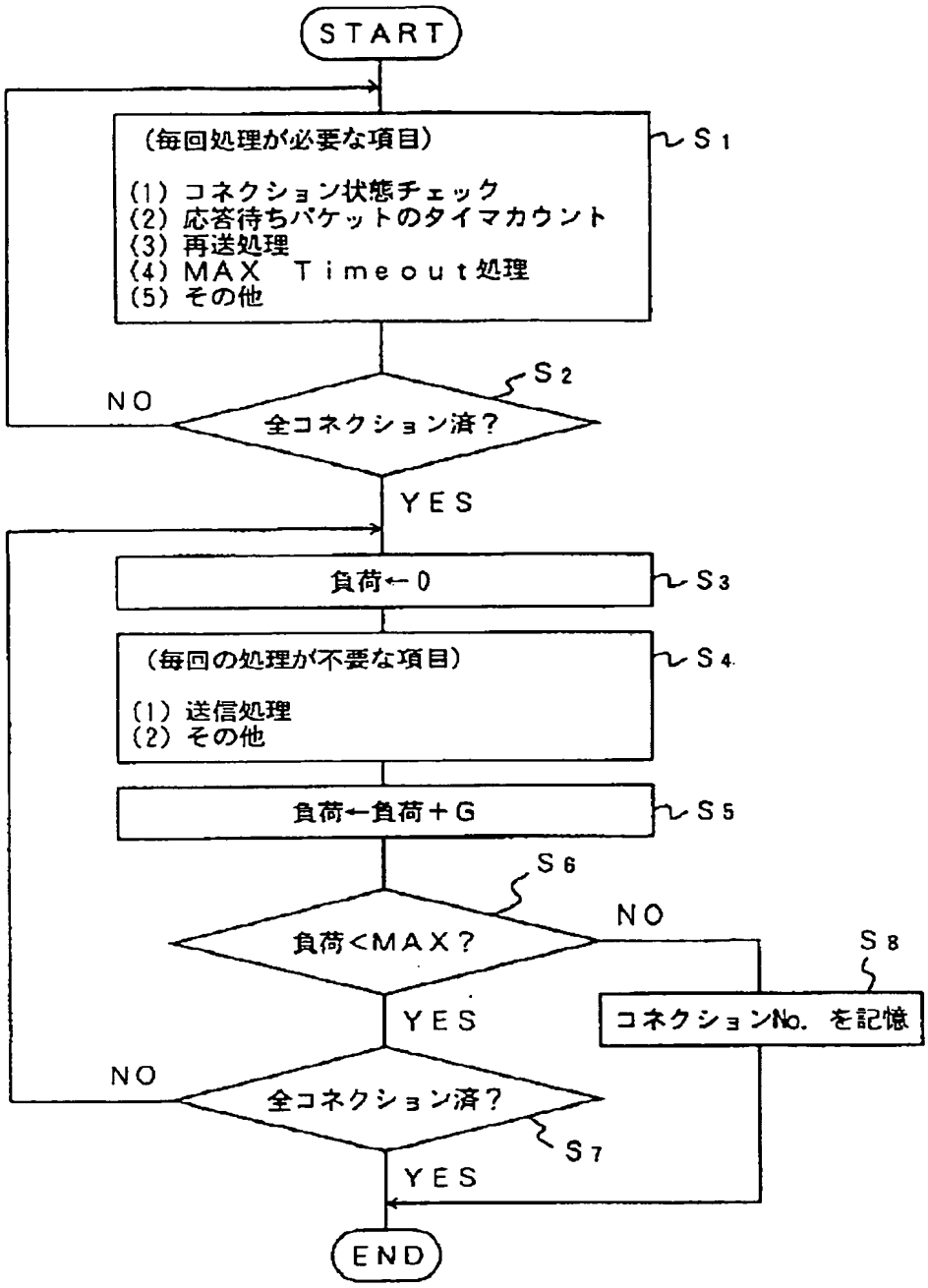
第４の実施の形態による負荷監視処理のブロック図

【図１６】



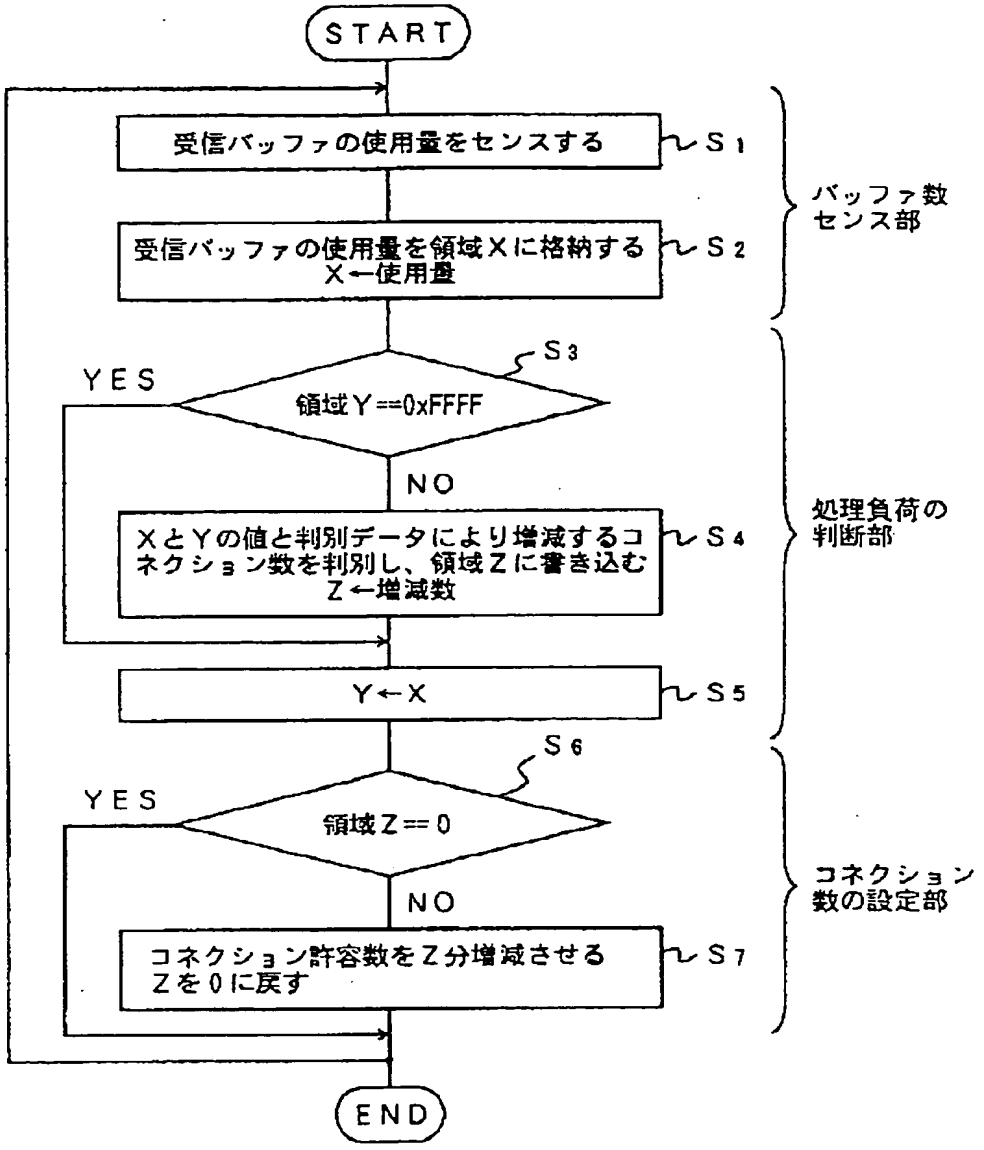
第4の実施の形態の動作を示すフローチャート

【図１０】



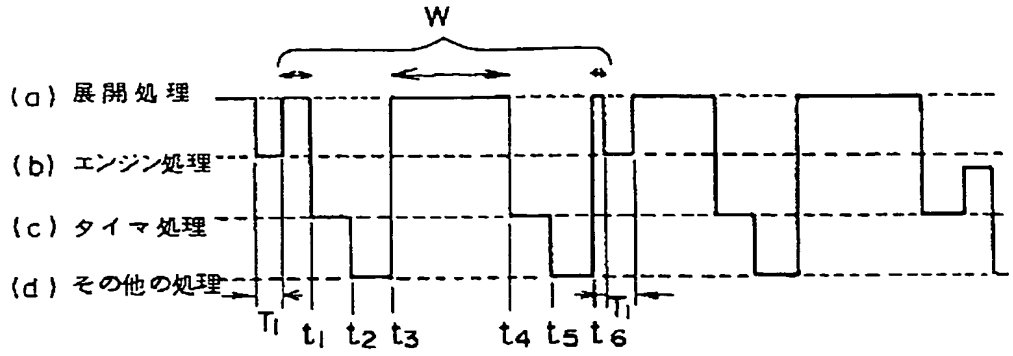
パケットを再送する際の処理を示すフローチャート

【図１３】



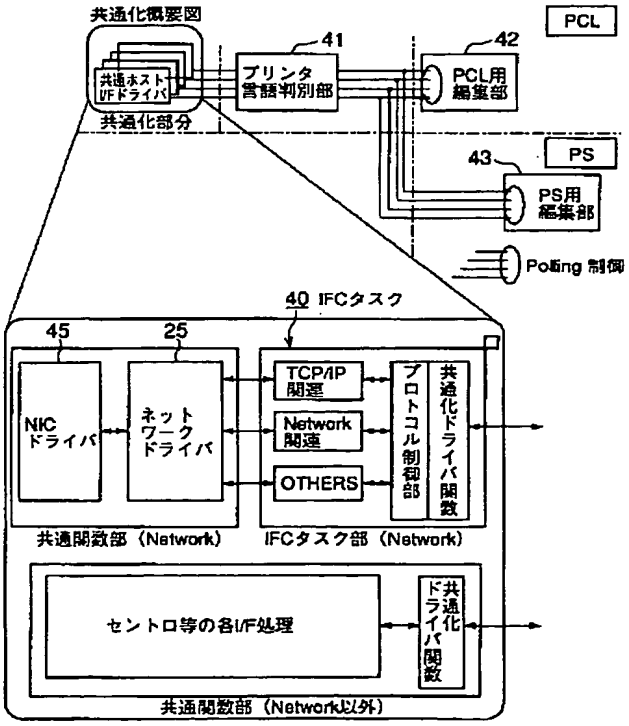
第３の実施の形態の動作を示すフローチャート

【図１５】



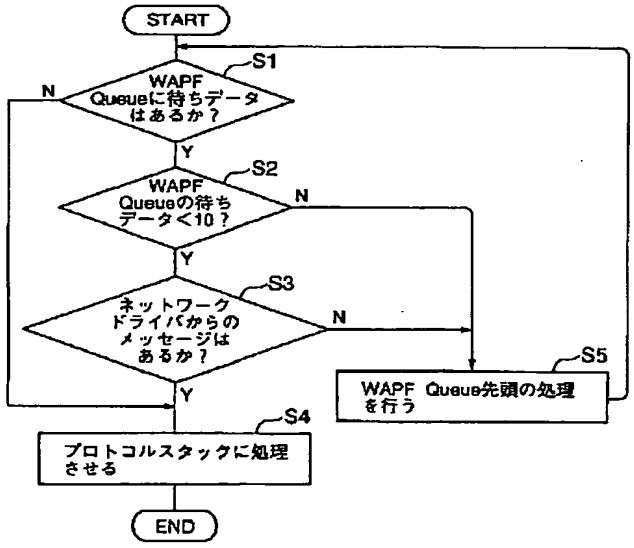
プリンタにおける展開処理とエンジン処理等とのタイムチャート

【図１７】



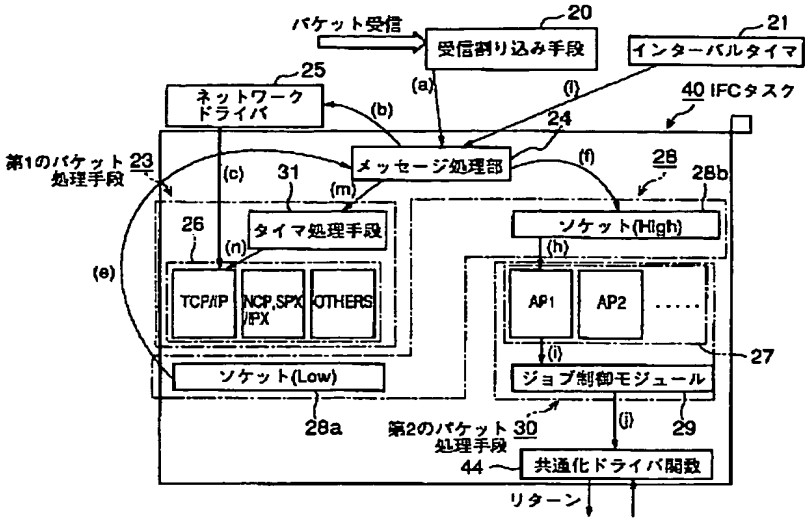
第5の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図

【図２１】



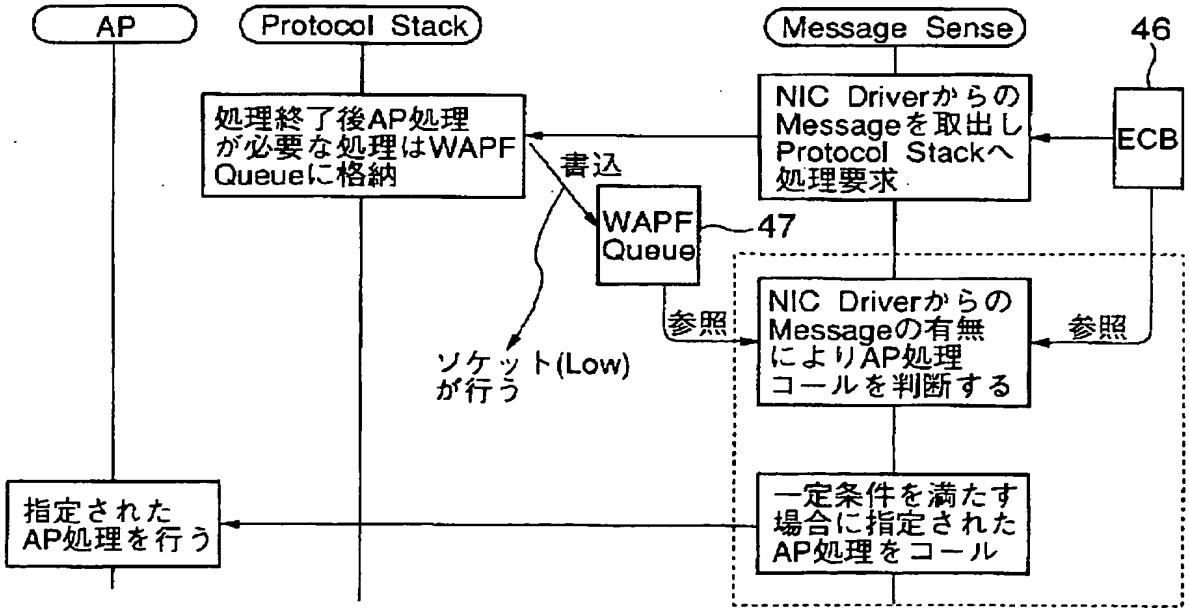
メッセージ処理手段の動作を示すフローチャート

【図１８】



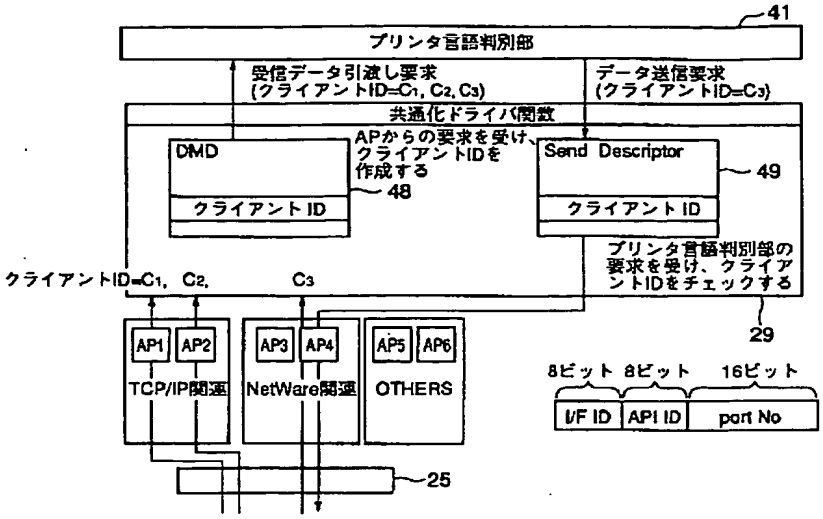
IFCタスクの構造を示すブロック図

【図２０】



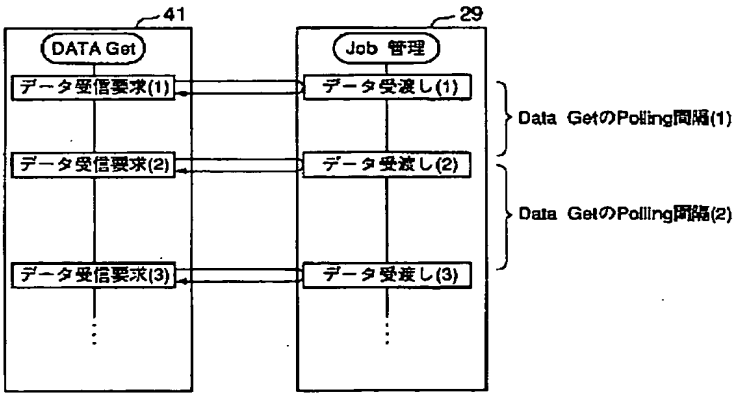
第6の実施の形態を示すブロック図

【図２２】



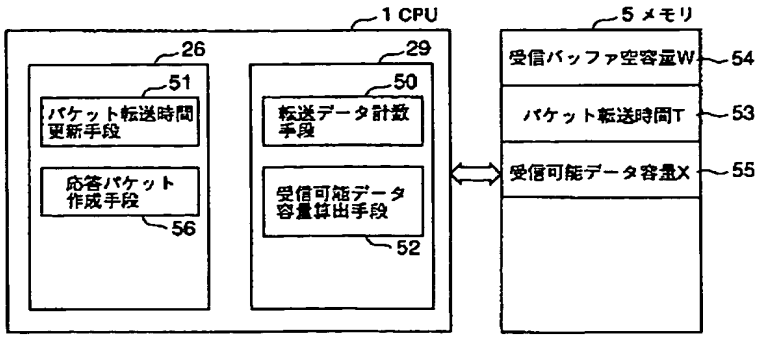
第7の実施の形態を示すブロック図

【図２３】



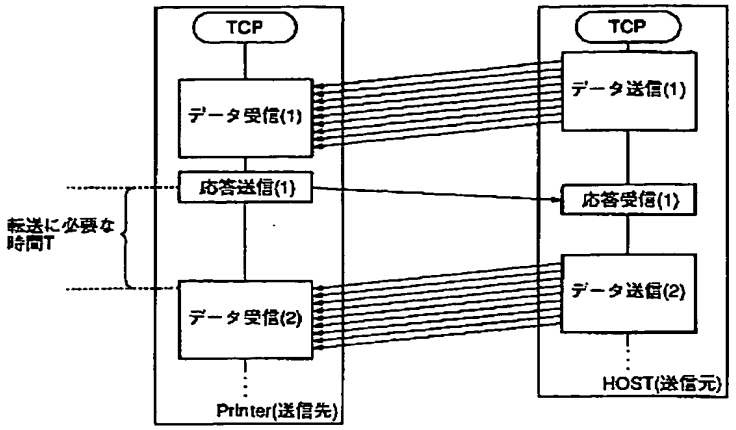
プリンタ言語判別部とジョブ制御モジュールとの間のデータ受け渡し処理の説明図

【図２５】



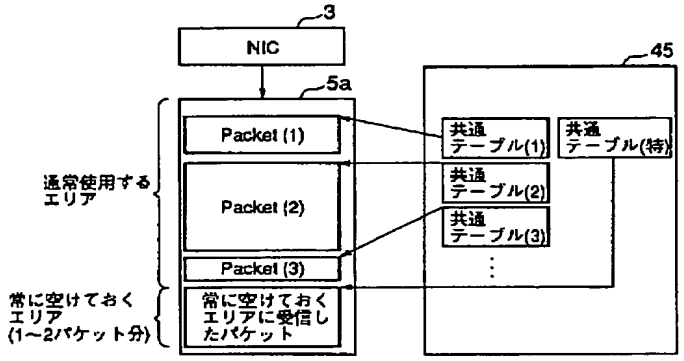
第8の実施の形態を示すブロック図

【図２４】



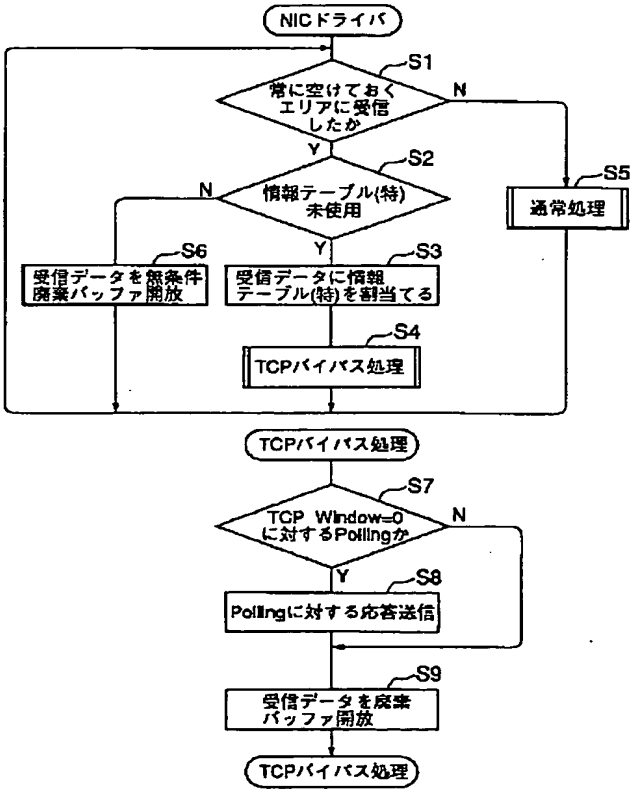
TCP間のデータ受け渡し処理の説明図

【図２６】



第9の実施の形態を示すブロック図

【図２７】



第9の実施の形態の動作を示すフローチャート